

Current Natural Sciences

Hui LIU

Time Series Predictive Control in Robotics



SCIENCE PRESS



Printed in France

EDP Sciences – ISBN(print): 978-2-7598-3509-6 – ISBN(ebook): 978-2-7598-3510-2
DOI: 10.1051/978-2-7598-3509-6

All rights relative to translation, adaptation and reproduction by any means whatsoever are reserved, worldwide. In accordance with the terms of paragraphs 2 and 3 of Article 41 of the French Act dated March 11, 1957, “copies or reproductions reserved strictly for private use and not intended for collective use” and, on the other hand, analyses and short quotations for example or illustrative purposes, are allowed. Otherwise, “any representation or reproduction – whether in full or in part – without the consent of the author or of his successors or assigns, is unlawful” (Article 40, paragraph 1). Any representation or reproduction, by any means whatsoever, will therefore be deemed an infringement of copyright punishable under Articles 425 and following of the French Penal Code.

The printed edition is not for sale in Chinese mainland. Customers in Chinese mainland please order the print book from Science Press. ISBN of the China edition: Science Press 978-7-03-078259-5

© Science Press, EDP Sciences, 2024

Preface

With the expansion of the robot market, the working environment of robots is diverse and complex. Various environmental challenges have profound effects on robot perception, navigation, planning, and control. With the development of artificial intelligence technology, intelligent autonomous robots have attracted attention, which have environmental adaptation and self-learning capabilities to complete more complex and dangerous tasks. In the current form of rapid development of the intelligent autonomous robotic system, it is necessary to publish an international book on planning and predictive control applications in smart robots. Therefore, the author completed the writing of this book based on their decades of research achievements and practical experience in the fields of data science and smart robots.

Based on the big data generated during the operation of smart robots, this book originally proposes the time series predictive control theory of smart robots to achieve high-precision control and application of various smart robots. This book focuses on the three mainstream robot configurations: mobile robot, robot arm, and wearable robot. The decomposition, modeling, prediction, and control of the navigation position, onboard power, motion, and other key elements that affect the application performance of the robot are carried out to provide readers with a detailed and complete theoretical basis and practical guidance.

This book provides the basic principles and advanced knowledge of data science and predictive control methods in the field of smart robotics. It plays an important role in enhancing the industrial competitiveness of the field of smart robotics and is of great significance in improving the intelligence of robots in autonomous decision-making, planning, and control. This book integrates a variety of cutting-edge theories and technologies, including big data, robotics, and artificial intelligence, and provides an important reference for future interdisciplinary collaborative development in the fields of robotics, intelligent manufacturing, automation science, and computer science.

The studies in the book are supported by the National Key R&D Program of China, the National Natural Science Foundation of China, the Hunan Province Science and Technology Talent Support Project, China, and the Changsha Science & Technology Project, China. In the process of writing the book, Ms. Shi Yin, Mr. Yu Xia, Mr. Zhu Duan, Mr. Yinan Xu, Mr. Yamin Fang, Ms. Jing Tan, Mr. Mengshuai Su, and other team members did a lot of work including data processing, code design and experimental tests. In addition, our sincere appreciation also goes to Prof. Dr. -Ing. habil. Kerstin Thurow from University of Rostock, Prof. Dr. Hans F. Fuchs and Dr. rer. nat. Yue Zhao from the University of Cologne for their precious help and support for this book. The authors would like to express heartfelt appreciation.

Prof. Dr. -Ing. habil. Hui Liu
Changsha, China
October 2022

Abbreviations

AdaBoost	Adaptive Boosting
ADF	Augmented Dickey-Fuller
AE	Autoencoder
AFSA	Artificial Fish Swarm Algorithm
AIC	Akaike Information Criterion
ANFIS	Adaptive Network-based Fuzzy Inference System
ANN	Artificial Neural Network
AR	Autoregressive
ARIA	American Industrial Robot Association
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
BGA	Binary Genetic Algorithm
BIC	Bayesian Information Criterion
CART	Classification and Regression Tree
CNN	Convolutional Neural Network
CWC	Coverage Width-based Criterion
DBN	Deep Belief Network
DT	Decision Tree
ECG	Electrocardiogram
EEG	Electroencephalogram
ELM	Extreme Learning Machine
EMG	Electromyogram
ENN	Elman Neural Network
EOG	Electrooculogram
ESN	Echo State Network
EWT	Empirical Wavelet Transform
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GBDT	Gradient Boosting Decision Tree

GMDH	Group Method of Data Handling
GNSS	Global Navigation Satellite System
GRNN	Generalized Regression Neural Network
GRU	Gated Recurrent Unit
GWO	Grey Wolf Optimization
HAL	Hybrid Assistive Limb
IMU	Inertial Measurement Unit
KF	Kalman Filter
KNN	K-Nearest Neighbor
LSTM	Long Short-Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ME	Mean Error
MLP	Multi-Layer Perceptron
MPC	Model Predictive Control
NAR	Nonlinear Autoregressive
NB	Naive Bayes
PCA	Principal Component Analysis
PICP	Prediction Interval Coverage Probability
PINAW	Prediction Interval Normalized Average Width
PRM	Probabilistic Road Map
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RELM	Regularized Extreme Learning Machine
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RTK	Real-Time Kinematic
SARIMA	Seasonal Autoregressive Integrated Moving Average
sEMG	Surface Electromyogram
SSA	Singular Spectrum Analysis
SVM	Support Vector Machine
SVR	Support Vector Regression
VMD	Variational Mode Decomposition
WPD	Wavelet Packet Decomposition
XGBoost	Extreme Gradient Boosting

Contents

Preface	III
Abbreviations	V
CHAPTER 1	
Introduction	1
1.1 Robotics and Control Technology	1
1.1.1 Robotics	1
1.1.2 Robotics Control Technology	4
1.2 Time Series Forecasting in Robotics Control	5
1.2.1 Time Series Forecasting Objectives	5
1.2.2 Time Series Forecasting Methods	8
1.3 Predictive Control in Robotics	10
1.3.1 Uncertainty Problems in Predictive Control of Robotics	10
1.3.2 Model Predictive Control	13
1.3.3 Significance and Purpose of Research	14
1.4 Scope of This Book	15
References	18
CHAPTER 2	
Robot Navigation Position Time Series Predictive Control	23
2.1 Introduction	23
2.2 Robot Navigation Position Time Series Measurement	24
2.3 Robot Navigation Position Time Series Uncertainty Analysis	25
2.4 Robot Navigation Position Time Series Statistical Forecasting Method	25
2.4.1 ARIMA Forecasting Algorithm	26
2.4.2 ARIMA-GARCH Forecasting Algorithm	30
2.5 Robot Navigation Position Time Series Intelligent Forecasting Method	35
2.5.1 RBF Neural Network Forecasting Algorithm	35
2.5.2 Elman Neural Network Forecasting Algorithm	38
2.5.3 Extreme Learning Machine Forecasting Algorithm	41

2.6 Robot Navigation Position Time Series Deep Learning Forecasting Method 44

2.6.1 LSTM Deep Neural Network Forecasting Algorithm 45

2.6.2 ESN Deep Neural Network Forecasting Algorithm 48

2.7 Comparative Analysis of Forecasting Performance 51

2.8 Robot Anti-Collision Monitoring and Control Based on Navigation Position Forecasting 52

2.9 Conclusions 53

References 53

CHAPTER 3

Mobile Robot Power Time Series Predictive Control 57

3.1 Introduction 57

3.2 Mobile Robot Power Time Series Measurement 58

3.3 Mobile Robot Power Time Series Uncertainty Analysis 59

3.4 Mobile Robot Power Time Series Statistical Forecasting Method 60

3.4.1 Experimental Design 60

3.4.2 Modeling Steps 61

3.4.3 Forecasting Results 63

3.5 Mobile Robot Power Time Series Intelligent Forecasting Method 64

3.5.1 Experimental Design 65

3.5.2 Modeling Steps 68

3.5.3 Forecasting Results 70

3.6 Mobile Robot Power Time Series Deep Learning Forecasting Method 71

3.6.1 Experimental Design 71

3.6.2 Modeling Steps 73

3.6.3 Forecasting Results 76

3.7 Comparative Analysis of Forecasting Performance 78

3.7.1 Analysis of Statistical Methods 78

3.7.2 Analysis of Intelligent Methods 78

3.7.3 Analysis of Deep Learning Methods 79

3.8 Mobile Robot Delivery Process Control Based on Power Forecasting 80

3.9 Conclusions 80

References 81

CHAPTER 4

Robot Arm Time Series Predictive Control 83

4.1 Introduction 83

4.2 Robot Arm Time Series Measurement 84

4.3 Robot Arm Time Series Uncertainty Analysis 85

4.4 Robot Arm Time Series Statistical Forecasting Method 85

4.4.1 Pandit–Wu Forecasting Algorithm 86

4.4.2 KF-ARMA Forecasting Algorithm 88

4.5 Robot Arm Time Series Intelligent Forecasting Method 93

4.5.1 RELM Forecasting Algorithm 93

4.5.2	XGBoost Forecasting Algorithm	97
4.5.3	GRNN Forecasting Algorithm	101
4.6	Robot Arm Time-Series Deep Learning Forecasting Method	104
4.6.1	Autoencoder Deep Neural Network Forecasting Algorithm	104
4.6.2	Deep Belief Network Forecasting Algorithm	107
4.7	Comparative Analysis of Forecasting Performance	110
4.7.1	Analysis of Statistical Methods	110
4.7.2	Analysis of Intelligent Methods	111
4.7.3	Analysis of Deep Learning Methods	111
4.8	Robot Arm Positioning Control Based on Arm Forecasting	112
4.9	Conclusions	113
	References	113

CHAPTER 5

	<u>Unmanned Vehicle Time Series Predictive Control</u>	<u>115</u>
5.1	Introduction	115
5.2	Unmanned Vehicle Time Series Measurement	118
5.3	Unmanned Vehicle Time Series Uncertainty Analysis	119
5.4	Unmanned Vehicle Time Series Statistical Forecasting Method	119
5.4.1	Kalman Filter Forecasting Algorithm	119
5.4.2	Fuzzy Time Series Forecasting Algorithm	122
5.5	Unmanned Vehicle Time Series Intelligent Forecasting Method	124
5.5.1	Elman Neural Network Forecasting Algorithm	125
5.5.2	NAR Neural Network Forecasting Algorithm	128
5.5.3	ANFIS Neural Network Forecasting Algorithm	130
5.6	Unmanned Vehicle Time Series Deep Learning Forecasting Method	134
5.6.1	RNN Deep Neural Network Forecasting Algorithm	134
5.6.2	LSTM Deep Neural Network Forecasting Algorithm	137
5.6.3	GRU Deep Neural Network Forecasting Algorithm	139
5.7	Comparative Analysis of Forecasting Performance	141
5.7.1	Analysis of Statistical Methods	141
5.7.2	Analysis of Intelligent Methods	142
5.7.3	Analysis of Deep Learning Methods	142
5.8	Unmanned Vehicle Navigation Control Based on Multi-Source Position Time Series Fusion	142
5.8.1	Unmanned Vehicle Fusion Positioning	142
5.8.2	Unmanned Vehicle Navigation Control	144
5.9	Unmanned Vehicle Charging Control Based on Multi-Source Power Time Series Fusion	145
5.10	Conclusions	146
	References	147

CHAPTER 6

Wearable Assistive Robot Time Series Predictive Control	151
6.1 Introduction	151
6.2 Wearable Assistive Robot Time Series Measurement	152
6.3 Wearable Assistive Robot Time Series Uncertainty Analysis	154
6.4 Wearable Assistive Robot Time Series Statistical Forecasting Method	155
6.4.1 Experimental Design	155
6.4.2 Modeling Step	160
6.4.3 Forecasting Results	162
6.5 Wearable Assistive Robot Time Series Intelligent Forecasting Method	165
6.5.1 Experimental Design	165
6.5.2 Modeling Step	167
6.5.3 Forecasting Results	171
6.6 Wearable Assistive Robot Time-Series Deep Learning Forecasting Method	174
6.6.1 Experimental Design	174
6.6.2 Modeling Step	176
6.6.3 Forecasting Results	179
6.7 Comparative Analysis of Forecasting Performance	180
6.8 Wearable Assistive Robot Motion Control Based on Forecasting	181
6.9 Conclusions	182
References	183

CHAPTER 7

Intelligent Manufacturing Performance Prediction and Application	187
7.1 Introduction	187
7.2 Data Acquisition	189
7.2.1 Data-Driven Method	190
7.2.2 Model-Driven Method	190
7.3 Prediction Modeling	193
7.3.1 Regression Algorithms	193
7.3.2 Artificial Neural Network (ANN)	199
7.3.3 Comparison Analysis	202
7.4 Application	204
7.4.1 System Configuration	204
7.4.2 The Other Application	216
7.5 Conclusions	217
References	218

Chapter 1

Introduction

1.1 Robotics and Control Technology

Since the middle of the 20th century, robotics has matured along with the rapid development of mechatronics, sensing technology, computer technology, and control theory. Robots have assisted or replaced human work in different scenarios. The emergence of new technologies such as 5G networks, cloud computing, and artificial intelligence in Industry 4.0 has brought about a new era of robotics development. Recently, China has become the world's largest market for robotics applications, and its market size of industrial, service, and specialized robots has grown steadily.

Robots were developed worldwide in the mid-20th century. They had simple structures, low control accuracy, and limited functions. Therefore, they could only serve humans to a certain extent and hardly perform complex operational tasks independently. Robots at this time fell far short of the expectations of scholars, prompting them to invest more research efforts in this area. By the 21st century, robotics has become relatively mature, with perfect functional composition, adaptation to various complex environments, and high mechanical control accuracy. Currently, the emergence of new technologies such as time-series prediction, human-robot interaction, and pattern recognition has brought new opportunities for robotics.

The predictive perception capability of robots requires improvement to overcome the influence of the work environment with complex uncertainty. Time series prediction technology is used to predict the trajectory, electrical energy, and joint coordinates of robots and performs feedback controls on the robot, realizing the functions of active obstacle avoidance, intelligent scheduling planning, and early error correction. Consequently, robot predictive control can improve their intelligence, reliability, and accuracy, which is crucial for building automated smart cities and improving human life qualities.

1.1.1 *Robotics*

The term robot first appeared in 1920, and it was not until 1940 that the science fiction writer Asimov gave a concrete definition of the concept of robots. Robot research began in the middle of the 20th century, and three generations of robots

have evolved (Li *et al.*). The first-generation robots were teaching robots. The second-generation robots are sensory robots equipped with sensors to adapt to certain changes in the external environment, such as sweeping robots. Intelligent robots are the third generation, which obtains external information through sensors and measuring equipment, use intelligent technology to identify and judge, and make corresponding planning decisions (Fan and Destech Publicat, 2016). It is an advanced robot that can act autonomously to achieve predetermined goals. Figure 1.1 shows the development history of robots.

In the 21st century, robots have been commonly used in industrial production, life services, and hazardous operations in human society, being of great importance to human production activities. Many countries and regions are committed to the development of robotics. Robotics is the science of designing, manufacturing and applying robots, and it consists of robot structure, kinematics, dynamics, control, machine perception, machine vision, robot language, decision-making, and planning (Brogardh, 2009). The current research hotspot aims at fully using big data technologies to study and control robot systems to help build an intelligent human society.

The history of robot development is shown in figure 1.1. Besides technological innovation, modern robot research also focuses on improving interactivity and adaptability, considering the human and environmental factors that interfere with it, which has also given birth to a new concept-robot system. The robot system comprises robots, work objects, and environments, including mechanical, drive, perception, and control systems (Angeles *et al.*, 2003). The robot system is a typical mechatronics system that is highly unified, intelligent, and flexible. It can achieve highly interactive work between the robot and the environment by applying advanced sensing, signal processing, and intelligent control technologies. The robot system is dedicated to helping robots complete operations under complex working conditions with higher efficiency and improving their perception, planning, and collaboration capabilities. Therefore, it improves human industrial production and brings convenience to human life.

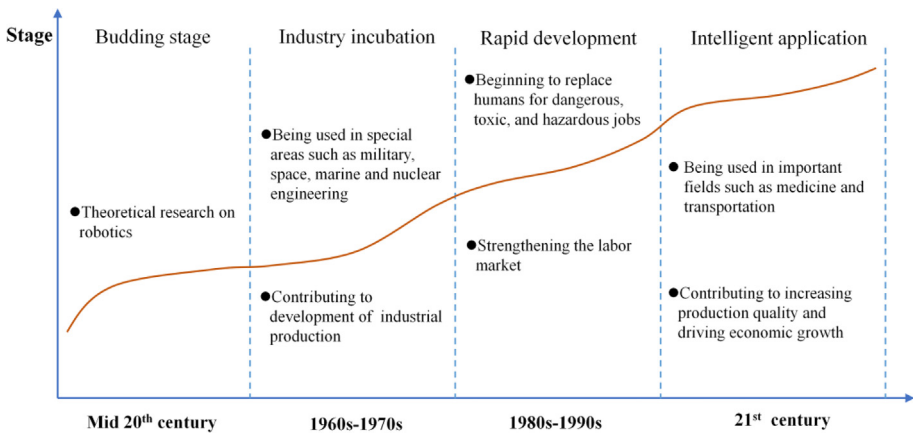


FIG. 1.1 – History of robot development.

Robots are technologically advanced, highly integrated, and rich in variety. Therefore, the classification of robots is very complex. No unified classification standard has been prescribed in previous studies (Dobra, 2015). This book attempts to cover a more comprehensive field and presents a rational classification based on four criteria, including robot control model, application scenario, working mode, and kinematics, as shown in figure 1.2.

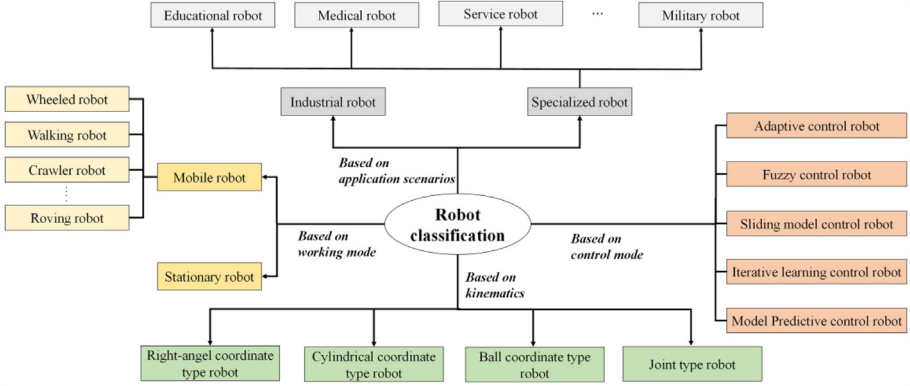


FIG. 1.2 – Robot classification.

A. Control mode

Robots can be classified into five categories according to their control methods: adaptive control, sliding model control, fuzzy control, iterative learning control, and model predictive control (Wu *et al.*, 2022). These controls are possible in both joint space and Cartesian space, with the corresponding physical quantity information transmitted by sensor acquisition. The control in joint space is for the variables at each joint of the robot, and the control in Cartesian space is for the variables at the end of the robot.

B. Application scenario

The application scenarios from robots can be divided into two main categories: industrial robots and specialty robots (Hodges and Hallam, 1990). This is widely recognized by scholars internationally. Industrial robots are robots with robotic arms or multiple degrees of freedom for industrial applications. Specialized robots are generally used in professional fields to serve, assist, or replace humans in performing tasks, such as service, medical, and military robots.

C. Working model

With the development of robotics, the functions of stationary robots are difficult to meet the increasingly diverse needs of humans. In the late 1980s, many countries began to focus on mobile robots with autonomous planning, self-organization,

self-adaptive capabilities, and the ability to adapt to complex unstructured environments (Siegwart *et al.*, 2011). They can be divided into wheeled, walking, crawling, and swimming robots according to the movement mode.

D. Kinematics

Robots can be divided into four types according to the form of motion. The right-angle coordinate type can move along three right-angle coordinates. The cylindrical coordinate type can lift, turn, and telescope. The ball coordinate type can turn, pitch, and telescope. The joint type has multiple rotating joints (Coiffet and Chirouze, 1982).

1.1.2 Robotics Control Technology

With the rapid development of robotics and the various application scenarios, robots have been equipped with intelligent functions, such as sensing, planning, decision-making, and control, satisfying accuracy requirements in robot operations. For robots to accurately perform various tasks and actions, advanced control technologies are required for manipulation. As a key technology for computer systems, robot control technology includes a wide range from robot intelligence to motion control and self-service control, which is shown in figure 1.3. Much research has been done on robot control technology to propose robot control strategies and schemes, achieving accurate machine motion and reasonable control performance (Durmus *et al.*, 2009).

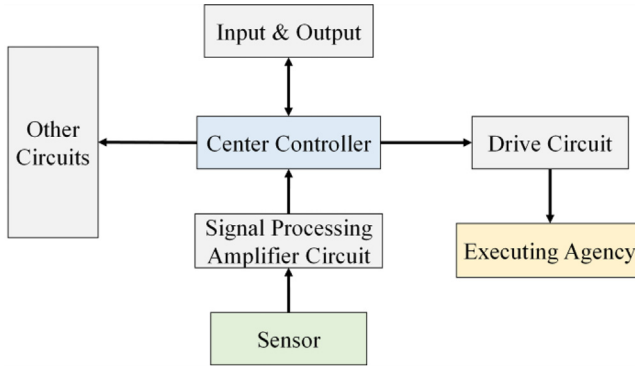


FIG. 1.3 – Robot control system structure.

The earliest robots were controlled by remote control, and the control actions were performed manually. Later, along with the development of computer technology, robots began to use computer systems to synthesize the functions of mechatronics devices and adopt the control method of demonstration and teaching. The recent development of information and control technologies has led to the

development of robot control technology toward intelligence, with new technologies such as off-line programming, task-level language, multi-sensor information fusion, and intelligent behavior control (Gerald and Feitian, 2020b). Intelligent robots obtain information about the environment through various sensors, use intelligent technologies to analyze it, make planning decisions, and finally achieve predetermined goals. The basic methods of existing robot control are divided into two types: Joint space control and Cartesian space control (Gerald and Feitian, 2020a). However, both types of control methods are susceptible to uncertainty, so more advanced and intelligent control techniques are needed to reduce errors in robot applications.

Artificial intelligence plays a key role in the development of future robots. The four major technologies of artificial intelligence, including memory technology, perception technology, action planning, and machine learning, have been fully developed in recent years, and these technologies can be transplanted into robots to form a complete, autonomous, and networked closed-loop control system (Ren and Bao, 2020). To solve the problems of low real-time control accuracy, susceptibility to environmental factors, and low fault tolerance of existing robot control technologies, time series prediction is a method with great potential. Many time series variables exist in the control process of robots, such as movement trajectory, electrical energy storage, joint displacement, and user intention. By deep information mining of historical data sequences, relevant potential laws can be obtained to accomplish complex control of robot trajectory planning, error correction, and human-robot interaction, promoting the development of intelligent robots.

1.2 Time Series Forecasting in Robotics Control

1.2.1 Time Series Forecasting Objectives

Time series forecasting is a powerful tool, and its importance has been proved by a large number of publications and extensive applications over the years (Tealab, 2018; De Gooijer and Hyndman, 2006). For example, using historical time series to predict the future state of wheeled mobile robots by obtaining their simulation models (Vagale *et al.*, 2021). As shown in figure 1.4, through time series prediction, the uncertainty of the robot system can be reduced enormously, and the modeling of unknown and complex items can be avoided. Consequently, the calculation cost can be greatly reduced. In addition, through the dynamic feedback structure of the predicted results, the robot's action and posture can be corrected in time, and the operation accuracy can be increased. Summarize the recent research and classify it based on the prediction objectives.

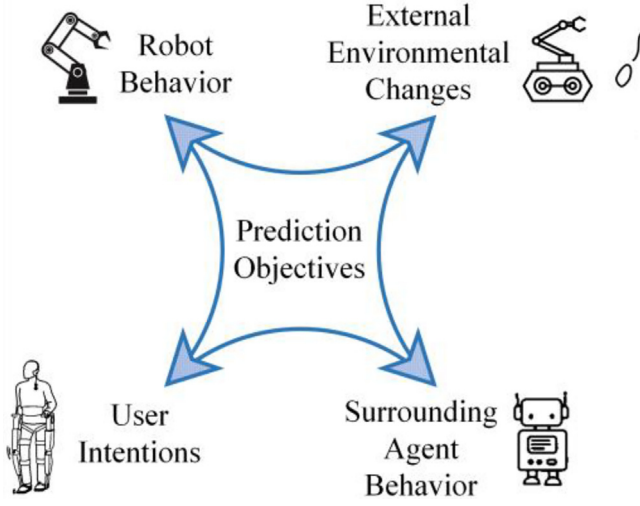


FIG. 1.4 – Objectives of time series prediction.

(1) Predicting robot behavior

Because the robot control simultaneously considers factors such as gravity load, attitude-related structural dynamics, and strong nonlinear interference of joint reducer, it is extremely difficult to predict the trajectory information using physical models (Chiu *et al.*, 2016). Therefore, the time series prediction method is required. The difference between the predicted and the actual trajectory can be analyzed by predicting the time series of robot behaviors. Then, the uncertainty of a robot system can be reduced, improving its moving accuracy. For example, a Temporal Convolutional Network (TCN) is used to predict the joint tracking error of industrial robots, and the pre-compensation method is adopted based on the prediction error to improve the joint tracking accuracy (Tan *et al.*, 2023).

In addition to predicting the motion trajectory of the robot's parts, time series prediction can also predict the power of the robot itself. Kerstin Thurow uses Wavelet Packet Decomposition (WPD) and Bidirectional Long Short Term Memory (BiLSTM) based on deep learning to predict the battery power and voltage of transport robots. The prediction results show the hybrid model's effectiveness and strong generalization ability for robot power prediction (Thurow *et al.*, 2019). There will be a sudden voltage drop in the early and late stages of battery discharge. However, in the middle stage of discharge, the battery voltage will change little, and the point of voltage drop will change with the use of the battery. Therefore, it is extremely difficult to calculate the battery power and battery level in the late stage of discharge. The traditional method is to warn when there is still some battery energy left and charge in advance. However, using time series prediction, we can know the battery level more accurately and reduce the number of charges while ensuring that the robot is not powered off.

(2) *Predicting external environment changes*

The robot system needs to interact with the surrounding physical world constantly. However, the physical world is complex and dynamic with unquantifiable items. The spatiotemporal evolution prediction for unknown dynamic information is essential for autonomous robot operation. The research shows that the heat convection–diffusion system and the fluid dynamical system can be effectively predicted by using the HybridNet integrated with a data-driven Deep Neural Network (DNN) and using Cellular Neural Network (CeNN) to solve model-driven calculations (Long *et al.*, 2018).

The prediction of external environmental changes can help robots respond to environmental changes in advance, such as avoiding falling rocks, collapsing rubble, and incoming floods. Researchers have already proved the feasibility of the Radial Basis Function (RBF) neural network. The RBF predicts the position of the dynamic obstacle at the next moment so that the avoidance problem of the dynamic obstacle can be converted into the avoidance problem of the static obstacle (Li *et al.*, 2008).

(3) *Predicting user intentions*

The wearable robot is an important part of the robot field. It is equipped on the user’s body and reacts according to the user’s intention. Predicting user intention through time series measurement can effectively avoid human–computer confrontation (HRC) in motion control, thus ensuring information interaction between humans and computers (Liu *et al.*, 2023). Unlike the previous physical-based human–computer cooperation, robots need to perceive the future actions of human beings in advance in prediction-based human–computer cooperation. This can improve the flexibility and smoothness of human–computer cooperation and reduce the security problems caused by HRC.

In addition to wearable robots, smart home robots also need to predict user intentions. In one study, multiple sensors were installed on different furniture to predict the furniture used in the next five minutes based on the furniture used in the past 45 min. This study also compared the prediction accuracy of multiple hybrid models based on Deep Belief Network (DBN) according to the prediction results (Choi *et al.*, 2013).

(4) *Predicting surrounding agent behavior*

The uncertainty of the robot system comes from other agents. For example, driverless vehicles need to predict the trajectory of surrounding vehicles or pedestrians to plan safe routes. However, unlike ordinary time series prediction, because the prediction object is another agent, they usually update their actions according to the actions of the predictor (Agarwal *et al.*, 2022). Therefore, the prediction of other agents can not only rely on historical time series but also take into account the content of the game theory.

1.2.2 Time Series Forecasting Methods

(1) Time series

There are time series in the fields of science and daily life, such as finance, physics, biology, medicine, meteorology, etc. Time series analysis is important to understand the objective world and natural phenomena. Generally speaking, the Autoregressive (AR) model proposed by British statistician George Udny Yule in 1927 is considered the origin of modern time series analysis (Yule, 1995; Nerlove, 1990).

The application of real-time online forecasting algorithms can better identify sensor information in an uncertain environment to improve the dynamic response frequency of the robot.

(2) Traditional statistical forecasting method

Traditional time series analysis methods mainly include descriptive time series analysis and time series analysis based on mathematical statistics (Aliev *et al.*, 2006). The descriptive analysis method mainly analyzes by directly comparing and observing the differences between the data or drawing images to explore the hidden laws between the sequences. It is characterized by convenient operation, intuitive, and efficient.

Traditional time series forecasting methods can be divided into three commonly used categories as follows:

- (a) Trend extrapolation forecasting technology (Martino, 1969). According to the existing and systematic mature methods to estimate the future data curve trend method. There are many kinds of trend extrapolation forecasting techniques, such as the Peel curve model and linear biexponential trend prediction model.
- (b) Regression prediction technology (Hoskuldsson, 1994). Regression forecasting technology is based on historical data. It is a forecasting method that establishes a regression equation through regression analysis. Scholars have studied various regression analysis techniques, such as linear regression, logistic regression, and least square regression. The regression model process is shown in figure 1.5.

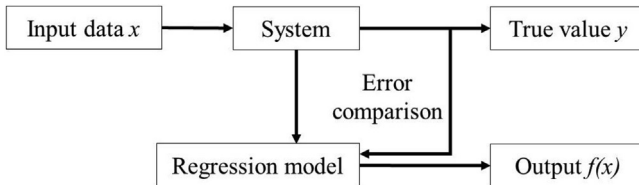


FIG. 1.5 – Regression model framework.

- (c) Grey forecasting technology (Xie and Wang, 2017; Deng, 1991). This technology is mainly aimed at variable data with uncertain factors and predicts future data. It is a type of model that is commonly used in small sample data. The advantage of this forecasting technology is that it can simplify complex forecasting problems into problems with exponential changes in data and establish and solve differential equations. Its disadvantage is that the prediction accuracy is unsatisfactory in the case of non-exponential changes in data.

The traditional forecasting method based on mathematical statistics is widely used even today. Its main idea is to take the historical data of time series as the observation value, use mathematical statistics to study the historical trajectory of the time series, and establish a model to fit future data. Commonly used Traditional statistical forecasting methods include the Autoregressive (AR) model, Moving Average (MA) models, Autoregressive Moving Average (ARMA) model, Autoregressive Integrated Moving Average (ARIMA) model, Naive Bayes (NB) model, Decision Tree, Hidden Markov Model (HMM), Random Forest (RF) model, etc.

(3) *Intelligent forecasting method*

Artificial neural networks were first introduced in 1943 (Huang *et al.*, 2009; McCulloch and Pitts, 1990). The structure of the artificial neural network is a model to simulate the information processing mechanism of the nervous system (Dorffner *et al.*). These networks can learn the complex relationship between their input and output vectors. So far, many different artificial neural networks, such as feedforward, recurrent, spiking, Kohonen self-organization, and radial basis function, have been expanded (Askari and Keynia, 2020; Ghosh-Dastidar and Adeli, 2009; Bebis and Georgiopoulos, 1994).

The advancement of artificial intelligence has brought people's attention to intelligent prediction methods. Because the intelligent model has a strong nonlinear fitting ability, its performance in classification and regression is better than the statistical model. Commonly used intelligent models include Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Extreme Learning Machine (ELM), Radial Basis Function (RBF), Elman Neural Network (ENN), Generalized Regression Neural Network (GRNN), etc.

The deep neural network with multiple hidden layers stacked can significantly enhance the learning ability and strengthen the feature extraction ability (Li, Zhang, and Ding, 2019; Li, Zhang, Ding, *et al.*, 2019). Because deep learning has a strong learning ability and performs well in many directions, many frameworks integrate relevant content. These frameworks are compatible with many platforms and have excellent portability. However, deep learning is highly dependent on data. The larger the amount of data, the better its performance. Especially in image recognition, facial recognition, NLP, and other fields. Therefore, only scientific research institutions or enterprises with relatively strong economic strength can use in-depth learning for cutting-edge and practical applications.

At present, Commonly used deep learning models include Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) network, Gated Recurrent Unit (GRU), Deep Belief Network (DBN), etc.

(4) *Hybrid forecasting method*

The main idea of the hybrid prediction method is to integrate various algorithms, methods, or models to make full use of their advantages, further improving the prediction performance. The prediction method can be combined with other algorithms or methods to optimize the prediction results at any stage.

Before the establishment of the basic prediction model, the data can be pre-processed, such as Wavelet Packet Decomposition (WPD), Empirical Wavelet Transform (EWT), Empirical Mode Decomposition (EMD), Variational Mode Decomposition (VMD), Factor Analysis (FA), Singular Spectrum Analysis (SSA), etc. The main purpose of pre-processing is to reduce noise interference, reduce the dimension of complex data, or extract data characteristics.

When building the model, various optimization algorithms can be used to optimize the model parameters, such as the number of hidden layers, learning rate, etc. Commonly used optimization algorithms include the Binary Genetic Algorithm (BGA), Grey Wolf Optimization (GWO), Particle Swarm Optimization (PSO), Simulated Annealing (SA) Algorithm, etc. These optimization algorithms can find better model parameters and improve generalization ability and prediction accuracy, which are especially helpful for models that are greatly affected by initial parameters.

After the model completes the prediction, the optimization method can also optimize the prediction results. For example, the error analysis method is used to analyze the error to correct the prediction error. In addition, ensemble learning can combine the prediction results of multiple weak learners according to a certain combination strategy. It obtains more significant generalization performance and higher prediction accuracy than a single learner.

1.3 Predictive Control in Robotics

1.3.1 *Uncertainty Problems in Predictive Control of Robotics*

Uncertainty Science is the product of the second scientific revolution in the last century (Evans and Rosenthal, 2004). As one of the active fields in human intelligence research, uncertainty science is a key technology for computer intelligence systems to become practical (Voorbraak, 1995). Uncertainty science studies systems with fuzzy structural concepts, incomplete information in calculations, unclear (black box) or incompletely clear (gray box) original data, energy dissipation, or material changes during the movement of things.

For mobile robots, the main content of uncertainty research is as follows: the abstract processing of uncertain knowledge or information, that is, how to effectively characterize data or information; the problem of multi-information integration and fusion, especially uncertainty information processing and fusion issues; image recognition and understanding in the uncertain environment of mobile robots,

real-time obstacle avoidance and path planning; the adaptability and robustness of reasoning models, and the error and performance analysis of different models; under uncertain environments, environment perception and environment modeling, and real-time positioning and navigation problems; the robot's self-reasoning and self-learning capabilities. Figure 1.6 analyzes the uncertain factors in mobile robots (Wang *et al.*, 2020; Scaglia *et al.*, 2015; Kim *et al.*, 2003).

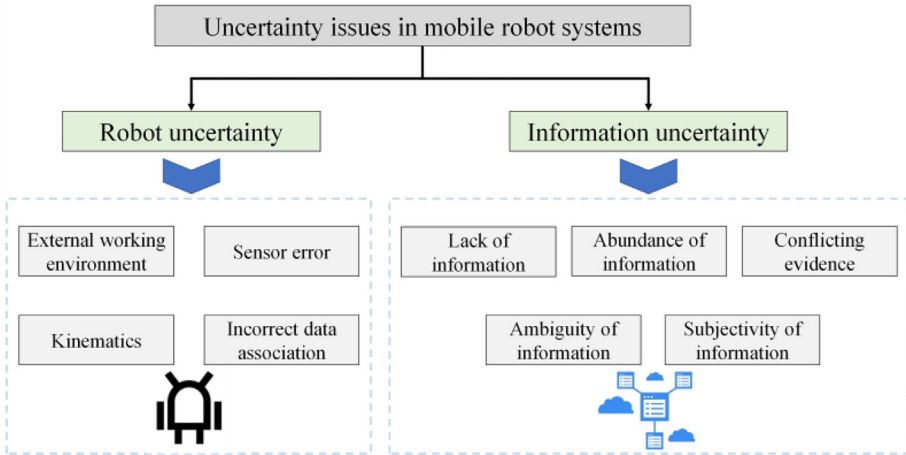


FIG. 1.6 – Uncertainty issues in the mobile robot system.

Uncertain factors in robot uncertainty are described as follows:

(a) Uncertain factors in the environment

The external working environment in which the robot is located is uncertain (Gasos and Rosetti, 1999). The existing algorithm research mostly assumes a static environment, while the real environment is constantly changing, such as the movement of people on the highway and the door opening in the home environment. These dynamic obstacles cause a series of errors due to the mismatch of map information during the robot's movement.

(b) Uncertain factors in sensors

The sensor of the robot system itself has accuracy errors in the measurement process, such as the cumulative track estimation error of the gyroscope, and it will also be affected by external interference factors, such as the influence of different surface characteristics and angular resolution of lidar (Chatila and Laumond, 1985). So, the perceived outside information contains much uncertainty.

(c) Uncertain factors in kinematics

The uncertainty that caused by the kinematics model (Santolaria and Gines, 2013). The mobile robot can obtain the current pose based on the kinematics model through sensor measurement. The kinematics model is a simplification of the real

world. It is generally built on flat terrain to predict the track of the mobile robot in a two-dimensional environment, so it is difficult to apply to complex terrain.

(d) Uncertain factors in data association

Errors are typically caused by incorrect data association. If m observations detected by sensors are used to update n features that already exist in the map, it is necessary to indicate the relationship between the features and observations and delete noisy data. The computational complexity of data association between m observations and the map with n features has an exponential relationship with m . The increase of environment complexity will increase m , and the increase of error will increase n . For some algorithms, inaccurate data association will lead to algorithm divergence. The process of uncertain motion information transmission is shown in figure 1.7.

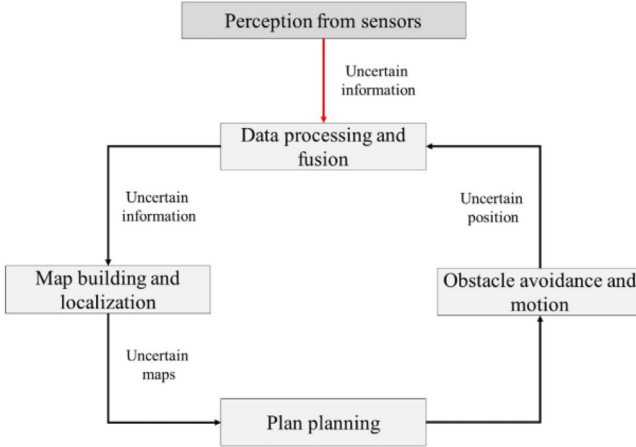


FIG. 1.7 – The process of uncertain motion information transmission.

In actual mobile robot applications, four types of information are often encountered: digital information, analog information, semantic information, and symbolic information. This information tends to show uncertainty which has the following reasons. A lack of information means that the description of a particular event or object is not comprehensive or complete, and the overall information cannot be obtained in the process of control (Xu *et al.*, 2019). For example, a sensor-based robot system cannot obtain complete distribution information about the environment. Therefore, the decision-making control cannot reach the ideal optimum, and there is a certain decision-making risk. The abundance of information means that the performance of information in structure and space has exceeded the direct processing capabilities of information processing mechanisms. The fusion and processing of multi-sensor information have become the primary problem that needs to be solved. Conflicting evidence is known as information contradiction. It means that the information obtained from different sources is contradictory, mutually

exclusive, and inconsistent. For example, in a multi-sensor system, some sensor information conflicts with each other due to environmental changes or other factors. The ambiguity of information is mainly manifested in the description of semantic information, that is, information description has multiple meanings and has a one-to-many mapping relationship. In essence, the ambiguity of information is due to the lack of complete information description and lack of information, which can also be classified as incomplete information. The subjectivity of information means that the meaning and importance of the same information will be different due to different objects (or observers) and occasions.

1.3.2 Model Predictive Control

In order to minimize the impact of uncertainties, it is crucial to study excellent robot control techniques. Model predictive control (MPC) theory was first introduced in the 1970s and has achieved a large number of applications in the field of engineering practice. Modern control theory has optimal performance indicators and systematic and accurate theoretical design methods. However, it has not yielded the expected results in industrial process control. The main reason is that modern control theory is based on accurate object parameter models, while industrial processes are often nonlinear, time-varying, strongly coupled, and uncertain, making it difficult to obtain accurate mathematical models and resulting in a significant decrease in control effectiveness (Norouzi *et al.*, 2023). To cope with the inconsistency between theoretical development and practical application, MPC, which is a method that can achieve high-quality control without high model accuracy requirements, has been explored from the characteristics and needs of industrial process control.

MPC consists of four main components (Camacho and Bordons, 1999), which are shown as follows:

(a) Predictive model

The core of predictive control lies in the prediction function, which can predict the future values of the process output based on the control inputs and historical information of the system at the present moment. The prediction model has the function of showing future dynamics. Therefore, it can develop control strategies based on the prediction results, compare the output changes under different strategies, and provide a basis for finding the optimal control strategy.

(b) Feedback correction

In predictive control, uncertainties such as nonlinearity, model mismatch, and disturbances in real situations make it impossible for model-based predictions to match reality accurately. However, in predictive control, the model's prediction error is obtained by comparing the output measurements with the model predictions. Consequently, the model can be corrected based on the prediction error. Because of the feedback correction process of the model, the predictive control has a strong ability to resist disturbances and overcome system uncertainty.

(c) Rolling optimization

Predictive control is an optimal control algorithm that requires optimizing a certain performance index to determine the future control role. However, the

optimization in predictive control differs from the usual discrete optimal control algorithm. This results from a rolling finite time domain optimization strategy used to replace an invariant global optimal objective. Its optimization process repeats online rather than offline once. At each sampling moment, the optimization performance metric involves only a finite time in the future from that moment to the next, and this optimization time period moves forward simultaneously to the next sampling moment.

(d) Reference track

In predictive control, to avoid sharp changes in the input and output of the process, the process output is often required to reach the set value along a desired, gentle curve, which is called the reference trajectory. The overview of model predictive control is shown in figure 1.8.

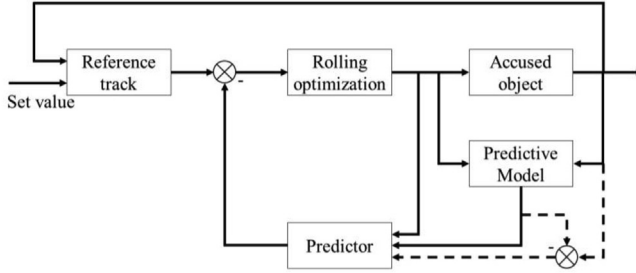


FIG. 1.8 – Overview of model predictive control.

Since MPC optimizes the current control policy over a certain time domain and considers constraints on state, input, and output variables, it is able to enforce constraints while performing optimization (Zhao *et al.*, 2023). This is a significant advantage in robotics application scenarios that are full of uncertainty. For example, after acquiring time series information such as the robot's motion trajectory, remaining electrical energy, and user's intention, the prediction model obtains prediction variables such as future action route, electrical energy change trend, and usage intention through feature extraction and deep learning. Then, it can formulate corresponding control strategies for robot positioning control, charging control, and anti-collision control based on future and current information.

1.3.3 Significance and Purpose of Research

The emergence of intelligent robots has attracted great attention from academia and industry and liberates people from repetitive, arduous, or dangerous tasks (Shit-sukane *et al.*, 2018). With the continuous advancement of technology, robot control technology has gradually matured. At the same time, the work scene of the robot also shows diversity and complexity, which poses a challenge to the robot's adaptability (Wang *et al.*, 2021). The trajectory control, electrical energy control,

and joint control for robots have attracted significant academic attention. In the case of robots with dynamic uncertainty in the working scenario, changes in environmental conditions, sensor failures, and randomness of obstacle movements may lead to disturbances in robot control (Li *et al.*, 2016). Especially in special operations and medical scenarios, the reliability of the robot control is required to be higher.

It is difficult for a robot to obtain an accurate representation of its surroundings and state directly from perceptual information. To deal with this uncertainty effectively, accurate robot control from limited perceptual information is an important issue. The robot control based on the time series prediction method is a novel, accurate, and efficient technique to eliminate the influence of uncertainty factors as much as possible by predicting relevant information in the future with existing reliable information.

The purpose of this book is to establish an effective time series prediction model to predict the robot movement trajectory, residual electrical energy variation, joint motion coordinates, and other information in future time by traditional prediction methods, intelligent prediction methods, and intelligent hybrid prediction methods based on the available data, and to compare and analyze the performance between different methods. Based on the prediction results, the robot is dynamically controlled by combining real-time information perception. The robot predictive control can achieve obstacle avoidance under uncertainty, power management, and scheduling under special working conditions, and early correction of deviation in case of sensing failure, contributing to the robustness and accuracy of robot control. It provides a reference for worldwide research related to robot predictive control.

1.4 Scope of This Book

As shown in figure 1.9, this book introduces the time series predictive control for robots, including robot anti-collision control, navigation control, charging control, and motion control. This book contains 7 chapters, which are introduced as follows:

Chapter 1: Introduction

Robotics is the science of studying, designing, manufacturing, and applying robots, including robot structure, coordinate system, kinematics, dynamics, control, machine perception, machine vision, robot language, decision-making, and planning. Chapter 1 begins with an introduction to the history of robotics, the current state of research, and future trends. Then the time series prediction methods in robotic systems are introduced, focusing on four types of prediction methods: statistical prediction methods, intelligent prediction methods, deep learning prediction methods, and hybrid prediction methods. Finally, the key problems of uncertainty in robotics predictive control and the related contents of model predictive control theory are briefly introduced, and the importance of predictive control research in this book is presented. Robotic predictive control can be applied to collision avoidance, navigation, charging, and motion control.

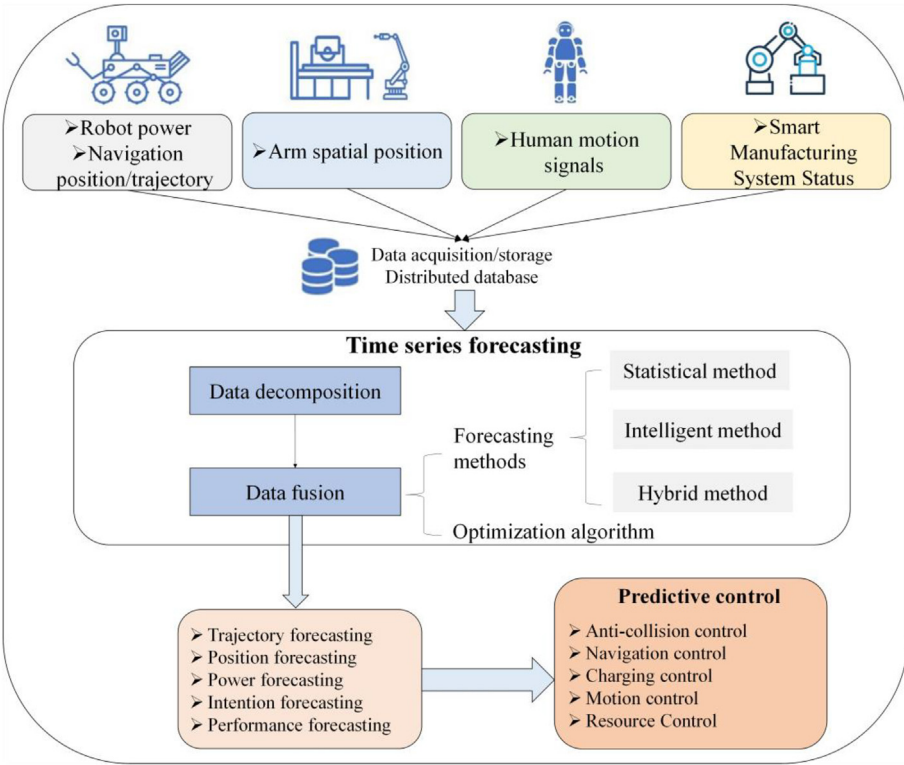


FIG. 1.9 – The scope of this book.

Chapter 2: Robot Navigation Position Time Series Predictive Control

Autonomous mobile robot has broad prospects for application in logistics, manufacturing, and commercial fields, which require robots to have autonomous intelligence in navigation and motion. Positioning and navigation are the basics of mobile robot applications. Chapter 2 illustrates several robot position forecasting models, including statistical methods, intelligent methods, and deep learning methods. The prediction results show that the intelligent model statistics model has better performance, and the use of decomposition, feature selection, and optimization algorithms can improve the prediction performance of the models. Position forecasting can help mobile robots interact with the environment, and achieve more robust anti-collision monitoring and control.

Chapter 3: Mobile Robot Power Time Series Predictive Control

The endurance of mobile robot is the most significant limitation in robot operation. Power management and health monitoring can guarantee safety and a continuous working state for mobile robot. Chapter 3 introduces the modeling and prediction methods of mobile robot power supply voltage time series based on a statistical model, hybrid intelligence model, and hybrid deep learning model.

The conclusion shows that smart and deep learning models have satisfactory accuracy in power and voltage forecasting, and time series decomposition is used to enhance the forecasting performance of these two methods. Battery power forecasting can help robot charge orderly and avoid production stoppages caused by the power failure of robot.

Chapter 4: Robot Arm Time Series Predictive Control

The robot arm is an important part of the robot working system. Chapter 4 models and analyzes the time series of the robot arm. Firstly, the measurement method and uncertainty of common manipulator time series are analyzed. Then the robot arm prediction models based on statistical models, intelligent models, and deep learning models are introduced respectively. The conclusion shows that intelligent models and statistical models perform better than deep learning models when the number of sequenced samples is small and the form of change is simple. Finally, the application of the robot arm time series prediction algorithm in industrial robots and laboratory robots is analyzed. The prediction of the robot arm time series can achieve more accurate robot arm control operation.

Chapter 5: Unmanned Vehicle Time Series Predictive Control

Unmanned vehicle is the main component of the intelligent transportation system, which can reduce traffic accidents, and ease traffic congestion and management pressure. Unmanned vehicles are in a complex dynamic environment involved with other traffic participants. Vehicle trajectory forecasting is a prerequisite for unmanned vehicle motion and interaction. Chapter 5 introduces the time series forecasting methods of vehicle trajectory in detail, including statistical forecasting methods, intelligent forecasting methods, and deep learning methods. The model results show that the deep learning algorithm can extract deeper information, so it has better robustness. Both the intelligent model and the deep learning model have satisfactory performance. Through the vehicle trajectory time series modeling and prediction, unmanned vehicle navigation control and charging control can be achieved.

Chapter 6: Wearable Assistive Robot Time Series Predictive Control

Wearable assistive robots can enhance the ability of wearers to perform tasks, effectively reduce labor costs and time costs, and assist the disabled or the elderly in physical rehabilitation. Chapter 6 introduces the time series forecasting methods of wearable assistive robots in detail, including statistical forecasting methods, intelligent forecasting methods, and deep learning methods, to recognize human movement intentions. The conclusion shows that the prediction results of the three types of algorithms are good, and the deep learning algorithms can achieve the best results. The accurate recognition of human movement intentions is the premise of wearable assistive robot motion control.

Chapter 7: Intelligent Manufacturing Performance Prediction and Application

A reasonable allocation of buffer capacity and machine processing time is necessary for intelligent manufacturing systems because of the unsynchronized

processing time of parts on different machines. Chapter 7 proposes a system performance prediction method based on data fusion technology, including regression algorithms and artificial neural networks, to solve the problem and develop a reasonable allocation scheme by performance prediction in manufacturing systems. The experimental results show that the regression algorithms outperform the artificial neural networks in the performance prediction of intelligent manufacturing systems. Therefore, they have good application prospects in the allocation of processing resources, storage resources, and transportation resources.

References

- Agarwal S., Fridovich-Keil D., Chinchali S. P. (2022) Robust forecasting for robotic control: A game-theoretic approach. *arXiv preprint* [arXiv:2209.10802](https://arxiv.org/abs/2209.10802).
- Aliev R., Fazlollahi B., Aliev R., Guirimov B. (2006) Fuzzy time series prediction method based on fuzzy recurrent neural network, *Neural information processing, pt 2, proceedings* (I. King, J. Wang, L. Chan, D. L. Wang, Eds). Vol. 4233, pp. 860–869.
- Alom M. Z., Taha T. M., Yakopcic C., Westberg S., Sidike P., Nasrin M. S., Hasan M., Van Essen B. C., Awwal A. A., Asari V. K. (2019) A state-of-the-art survey on deep learning theory and architectures, *Electronics* **8**(3), 292.
- Angeles J., Angeles J., SpringerLink. (2003) *Fundamentals of robotic mechanical systems: Theory, methods, and algorithms*, 2nd 2003 edn. Springer, New York: Imprint: Springer.
- Askari M., Keynia F. (2020) Mid-term electricity load forecasting by a new composite method based on optimal learning MLP algorithm, *IET Gener. Transm. Distrib.* **14**(5), 845–852.
- Barenji A. V., Barenji R. V., Hashemipour M. (2016) Flexible testing platform for employment of RFID-enabled multi-agent system on flexible assembly line, *Adv. Eng. Softw.* **91**, 1–11.
- Bebis G., Georgiopoulos M. (1994) Feed-forward neural networks, *IEEE Potentials* **13**(4), 27–31.
- Brogardh T. (2009) Robot control overview: An industrial perspective, *Model. Identif. Control* **30**(3), 167–180.
- Camacho E. F., Bordons C. (1999) Model based predictive controllers, *Model predictive control* (E. F. Camacho, C. Bordons, Eds). Springer, London, pp. 13–31.
- Chatila R., Laumond J. (1985, 25–28 March) Position referencing and consistent world modeling for mobile robots. Proceedings. 1985 IEEE International Conference on Robotics and Automation.
- Chen P., Pedersen T., Bak-Jensen B., Chen Z. (2009) ARIMA-based time series model of stochastic wind power generation, *IEEE Trans. Power Syst.* **25**(2), 667–676.
- Chiu S. H., Chen C. C., Chen K. T., Huang X. J., Pong S. H. (2016) Joint position-based impedance control with load compensation for robot arm, *J. Chin. Inst. Eng.* **39**(3), 337–344.
- Cho T.-H., Hwang H.-R., Lee J.-H., Lee I.-S. (2018) Comparison of intelligent methods of SOC estimation for battery of photovoltaic system, *Int. J. Adv. Comput. Sci. App.* **9**(9), 49–56.
- Choi S., Kim E., Oh S. (2013) *Human behavior prediction for smart homes using deep learning*. Ro-man.
- Coiffet P., Chirouze M. (1982) *An introduction to robot technology*, 1st 1983 edn. Kogan Page.
- Contreras J., Espinola R., Nogales F. J., Conejo A. J. (2003) ARIMA models to predict next-day electricity prices, *IEEE Trans. Power Syst.* **18**(3), 1014–1020.
- De Gooijer J. G., Hyndman R. J. (2006) 25 years of time series forecasting, *Int. J. Forecast.* **22**(3), 443–473.
- Deng J. (1991) Application of grey system theory in China. Proceedings. First International Symposium on Uncertainty Modeling and Analysis, 1990.
- DiGiampaolo E., Martinelli F. (2018) A robotic system for localization of passive UHF-RFID tagged objects on shelves, *IEEE Sens. J.* **18**(20), 8558–8568.

- Ding S., Xu X., Zhu H., Wang J., Jin F. (2011) Studies on optimization algorithms for some artificial neural networks based on genetic algorithm (GA), *J. Comput.* **6**(5), 939–946.
- Dobra A. (2015) General classification of robots. Size criteria. International Conference on Robotics in Alpe-adria-danube Region.
- Dorffner G. (2001) *Artificial Neural Networks-ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings* (Vol. 2130). Springer Science & Business Media.
- Dorffner G., Bischof H., Hornik K. (2001) Artificial neural networks – ICANN 2001. International conference, Vienna, Austria, August 21–25, 2001. Proceedings.
- Du K.-L., Swamy M. N. (2013) *Neural networks and statistical learning*. Springer Science & Business Media.
- Duan Z., Liu H. (2019) An evolution-dependent multi-objective ensemble model of vanishing moment with adversarial auto-encoder for short-term wind speed forecasting in Xinjiang wind farm, China, *Energy Convers. Manage.* **198**, 111914.
- Durmus B., Temurtas H., Yumusak N., Temurtas F. (2009) A study on industrial robotic manipulator model using model based predictive controls, *J. Intell. Manuf.* **20**(2), 233–241.
- Eliakim I., Cohen Z., Kosa G., Yovel Y. (2018) A fully autonomous terrestrial bat-like acoustic robot, *PLoS Comput. Biol.* **14**(9), e1006406.
- Elnagar A. (2011) Prediction of future configurations of a moving target in a time-varying environment using an autoregressive model, *Intell. Control Autom.* **2**(04), 284–292.
- Evans M., Rosenthal J. S. (2004) *Probability and statistics: The science of uncertainty*. W.H. Freeman and Co.
- Fan L. G., Destech Publicat I. (2016) *Research on Effect of Wind Power Accommodation by Synergistic Operation Based on Electric Boiler and Heat Accumulator*. 2016 International Conference on Electrical Engineering and Automation (ICEEA 2016).
- Ganjidoost H., Mousavi S. J., Soroush A. (2016) Adaptive network-based fuzzy inference systems coupled with genetic algorithms for predicting soil permeability coefficient, *Neural Process. Lett.* **44**(1), 53–79.
- Garcia R. C., Contreras J., Van Akkeren M., Garcia J. B. C. (2005) A GARCH forecasting model to predict day-ahead electricity prices, *IEEE Trans. Power Syst.* **20**(2), 867–874.
- Gasos J., Rosetti A. (1999) Uncertainty representation for mobile robots: Perception, modeling and navigation in unknown environments, *Fuzzy Sets Syst.* **107**(1), 1–24.
- Gerald C., Feitian Z. (2020a) Kinematic models for mobile robots, *Mobile robots: Navigation, control and sensing, surface robots and AUVs*. IEEE, pp. 5–12.
- Gerald C., Feitian Z. (2020b) Mobile robot control, *Mobile robots: Navigation, control and sensing, surface robots and AUVs*. IEEE, pp. 13–70.
- Gheys I. A., Smith L. S. (2011) A novel neural network ensemble architecture for time series forecasting, *Neurocomputing* **74**(18), 3855–3864.
- Ghosh-Dastidar S., Adeli H. (2009) Spiking neural networks, *Int. J. Neural Syst.* **19**(4), 295–308.
- Guo S., Fang T.-T., Song T., Xi F.-F., Wei B.-G. (2018) Tracking and localization for omni-directional mobile industrial robot using reflectors, *Adv. Manuf.* **6**(1), 118–125.
- Hodges B., Hallam P. (1990) *Industrial robotics*. Heinemann Newnes.
- Hol E. M. (2013) *Empirical studies on volatility in international stock markets* (Vol. 6). Springer Science & Business Media.
- Hoskuldsson A. (1994) Prediction methods in science and technology.: Vol. 1. Basic theory.
- Huang D. S., Heutte L., Loog M. (2009) Proceedings of the 3rd International Conference on Intelligent Computing: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence.
- Jia Y., Mo Y., Wang W., Guo S., Ma L. (2021) Deep-Sea Source Ranging Method Using Modified General Regression Neural Network. 2021 OES China Ocean Acoustics (COA).
- Khosravi A., Nahavandi S., Creighton D. (2013) Quantifying uncertainties of neural network-based electricity price forecasts, *Appl. Energy* **112**, 120–129.
- Kim M. S., Shin J. H., Hong S. G., Lee J. J. (2003) Designing a robust adaptive dynamic controller for nonholonomic mobile robots under modeling uncertainty and disturbances, *Mechatronics* **13** (5), 507–519.

- Kostavelis I., Charalampous K., Gasteratos A., Tsotsos J. K. (2016) Robot navigation via spatial and temporal coherent semantic maps, *Eng. App. Artif. Intell.* **48**, 173–187.
- Lathi B. P., Green R. A. (2005) *Linear systems and signals* (Vol. 2). Oxford University Press, New York.
- Li C., Tang X. (2016) Proceedings in Adaptation, Learning and Optimization. IES.
- Li C. H., Li Y. B., Wang F. Y., IEEE. (2008) *Research of the Obstacle Avoidance Based on RBFNN for the Mobile Robot Under Dynamic Environment*. 2008 7th World Congress on Intelligent Control and Automation, Vols. 1–23.
- Li K., Xue Y., Cui S., Niu Q., Yang Z., Luk P. (2017) *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration: International Conference on Life System Modeling and Simulation, LSMS 2017 and International Conference on Intelligent Computing for Sustainable Energy and Environment, ICSEE 2017, Nanjing, China, September 22–24, 2017, Proceedings, Part III* (Vol. 763). Springer.
- Li X., Zhang W., Ding Q. (2019) Understanding and improving deep learning-based rolling bearing fault diagnosis with attention mechanism, *Signal Process.* **161**, 136–154.
- Li X., Zhang W., Ding Q., Sun J. Q. (2019) Multi-layer domain adaptation method for rolling bearing fault diagnosis, *Signal Process.* **157**, 180–197.
- Li Z. J., Deng J., Lu R. Q., Xu Y., Bai J. J., Su C. Y. (2016) Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach, *IEEE Trans. Syst. Man Cybern.-Syst.* **46**(6), 740–749.
- Lian J., Dong P., Zhang Y., Pan J., Liu K. (2020) A novel data-driven tropical cyclone track prediction model based on CNN and GRU with multi-dimensional feature selection, *IEEE Access* **8**, 97114–97128.
- Lin X., Yang Z., Song Y. (2009) Short-term stock price prediction based on echo state networks, *Expert Syst. App.* **36**(3), 7313–7317.
- Liu H. (2020) *Smart cities: Big data prediction methods and applications*. Springer Nature.
- Liu H., Duan Z. (2020) Corrected multi-resolution ensemble model for wind power forecasting with real-time decomposition and Bivariate Kernel density estimation, *Energy Convers. Manage.* **203**, 112265.
- Liu H., Duan Z., Chen C. (2019) A hybrid framework for forecasting PM2. 5 concentrations using multi-step deterministic and probabilistic strategy, *Air Qual. Atmos. Health* **12**(7), 785–795.
- Liu H., Stoll N., Junginger S., Thurow K. (2015) *New localization strategy for mobile robot transportation in life science automation using StarGazer sensor, time series modeling and Kalman filter processing*. 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO).
- Liu H., Xu Y., Wu X., Lv X., Zhang D., Zhong G. (2019) Big data forecasting model of indoor positions for mobile robot navigation based on apache spark platform. 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA).
- Liu S. M., Brorsen B. W. (1995) Maximum likelihood estimation of a GARCH-stable model, *J. Appl. Econ.* **10**(3), 273–285.
- Liu Y., Liu K., Wang G., Sun Z., Jin L. (2023) Noise-tolerant zeroing neurodynamic algorithm for upper limb motion intention-based human-robot interaction control in non-ideal conditions, *Expert Syst. App.* **213**, 118891.
- Long Y., She X., Mukhopadhyay S. (2018) HybridNet: Integrating Model-based and Data-driven Learning to Predict Evolution of Dynamical Systems.
- Luo B., Miao S. (2019) A Literature Survey on Electricity Price Forecasting in Deregulated Markets. 2019 IEEE Sustainable Power and Energy Conference (iSPEC).
- Luo R. C., Hsiao T. J. (2019) Dynamic wireless indoor localization incorporating with an autonomous mobile robot based on an adaptive signal model fingerprinting approach, *IEEE Trans. Ind. Electron.* **66**(3), 1940–1951.
- Man Z., Lee K., Wang D., Cao Z., Miao C. (2011) A modified ELM algorithm for single-hidden layer feedforward neural networks with linear nodes. 2011 6th IEEE conference on industrial electronics and applications.
- Martino J. P. (1969) *An introduction to technological forecasting*. Assessing the Future & Policy Planning Gordon & Breach Science Publishers.

- McCulloch W. S., Pitts W. (1990) A logical calculus of the ideas immanent in nervous activity. 1943 [Journal Article], *Bull. Math. Biol.* **52**(1–2), 99–115; discussion 173–197.
- Moskalenko V., Moskalenko A., Korobov A., Semashko V. (2019) The model and training algorithm of compact drone autonomous visual navigation system, *Data* **4**(1), 4.
- Namba T., Yamada Y. (2018) Risks of deep reinforcement learning applied to fall prevention assist by autonomous mobile robots in the hospital, *Big Data Cognit. Comput.* **2**(2), 13.
- Nerlove M. (1990) *Autoregressive and moving-average time-series processes*. Palgrave Macmillan UK.
- Norouzi A., Heidarifar H., Borhan H., Shahbakhti M., Koch C. R. (2023) Integrating machine learning and model predictive control for automotive applications: A review and future directions, *Eng. App. Artif. Intell.* **120**, 105878.
- Onieva E., Santos I., Osaba E., Quintián H., Corchado E. (2015) *Hybrid Artificial Intelligent Systems: 10th International Conference, HAIS 2015, Bilbao, Spain, June 22–24, 2015, Proceedings* (Vol. 9121). Springer.
- Ren F. J., Bao Y. W. (2020) A review on human-computer interaction and intelligent robots, *Int. J. Inf. Technol. Decis. Making* **19**(1), 5–47.
- Ren L., Liu Y., Rui Z., Li H., Feng R. (2009) Application of Elman neural network and MATLAB to load forecasting. 2009 International Conference on Information Technology and Computer Science.
- Saeedvand S., Aghdasi H. S., Baltes J. (2018) Novel lightweight odometric learning method for humanoid robot localization, *Mechatronics* **55**, 38–53.
- Sanchez-Garcia A. J., Rios-Figueroa H. V., Garnier H., Quintana-Carapia G., Rechy-Ramirez E. J., Marin-Hernandez A. (2018) Predicting collisions: Time-to-contact forecasting based on probabilistic segmentation and system identification, *Adv. Rob.* **32**(8), 426–442.
- Santolaria J., Gines M. (2013) Uncertainty estimation in robot kinematic calibration, *Rob. Comput.-Integr. Manuf.* **29**(2), 370–384.
- Scaglia G., Serrano E., Rosales A., Albertos P. (2015) Linear interpolation based controller design for trajectory tracking under uncertainties: Application to mobile robots, *Control Eng. Pract.* **45**, 123–132.
- Schalkoff R. J. (2007) *Pattern recognition*. Wiley Encyclopedia of Computer Science and Engineering.
- Sheinker A., Ginzburg B., Salomonski N., Frumkis L., Kaplan B.-Z., Moldwin M. B. (2016) A method for indoor navigation based on magnetic beacons using smartphones and tablets, *Measurement* **81**, 197–209.
- Shitsukane A., Cheriuyot W., Otieno C., Mvurya M. (2018) A survey on obstacles avoidance mobile robot in static unknown environment, *Int. J. Comput.*
- Siegwart R., Nourbakhsh I. R., Scaramuzza D., Xplora I., Press M. I. T. (2011) *Introduction to autonomous mobile robots*, 2nd edn. MIT Press IEEE Xplore.
- Son D., Dong X., Sitti M. (2019) A simultaneous calibration method for magnetic robot localization and actuation systems, *IEEE Trans. Rob.* **35**(2), 343–352. <https://doi.org/10.1109/TRO.2018.2885218>.
- Tamura Y., Sato T., Ooe M., Ishiguro M. (1991) A procedure for tidal analysis with a Bayesian information criterion, *Geophys. J. Int.* **104**(3), 507–516.
- Tan S., Yang J., Ding H. (2023) A prediction and compensation method of robot tracking error considering pose-dependent load decomposition, *Rob. Comput.-Integr. Manuf.* **80**, 102476.
- Teal R., Tubana B., Girma K., Freeman K., Arnall D., Walsh O., Raun W. (2006) In-season prediction of corn grain yield potential using normalized difference vegetation index, *Agron. J.* **98**(6), 1488–1494.
- Tealab A. (2018) Time series forecasting using artificial neural networks methodologies: A systematic review, *Future Comput. Inf. J.* **3**(2).
- Thong-un N. (2016) A linear least square approach for navigation using ultrasonic waves, *ECTI Trans. Electr. Eng. Electron. Commun.* **14**(1), 65–74.
- Thurrow K., Chen C., Junginger S., Stoll N., Liu H. (2019) Transportation robot battery power forecasting based on bidirectional deep-learning method, *Transp. Saf. Environ.* **1**(3), 205–211.

- Torres-Solis J., Falk T. H., Chau T. (2010) A review of indoor localization technologies: Towards navigational assistance for topographical disorientation, *Ambient intelligence*. IntechOpen.
- Vagale A., Steina L., Vecins V. (2021) Time series forecasting of mobile robot motion sensors using LSTM networks, *Appl. Comput. Syst.* **26**(2), 150–157.
- Voorbraak F. (1995) Reasoning with uncertainty in AI. Reasoning with Uncertainty in Robotics, International Workshop, RUR '95, Amsterdam, The Netherlands, December 4–6, 1995, Proceedings.
- Wang C., Zhang X. H., Zang X. Z., Liu Y. B., Ding G. W., Yin W. X., Zhao J. (2020) Feature sensing and robotic grasping of objects with uncertain information: A review, *Sensors* **20**(13), Article 3707.
- Wang Y., Li X. X., Zhang J., Li S., Xu Z. H., Zhou X. F. (2021) Review of wheeled mobile robot collision avoidance under unknown environment, *Sci. Prog.* **104**(3), Article 00368504211037771.
- Wu J., Jin Z., Liu A., Yu L., Yang F. (2022) A survey of learning-based control of robotic visual servoing systems, *J. Franklin Inst.* **359**(1), 556–577.
- Xie N. M., Wang R. Z. (2017) A historic review of grey forecasting models, *J. Grey Syst.* **29**(4), 1–29.
- Xu J., Cooke F. L., Gen M., Ahmed S. E., SpringerLink. (2019) *Proceedings of the Twelfth International Conference on Management Science and Engineering Management* (1st edition 2019 edn). Springer.
- Xydas E., Qadrdan M., Marmaras C., Cipcigan L., Jenkins N., Ameli H. (2017) Probabilistic wind power forecasting and its application in the scheduling of gas-fired generators, *Appl. Energy* **192**, 382–394.
- Yamada Y., Ito K., Tsuji T., Otani K., Kobayashi R., Watanabe Y., Hiryu S. (2019) Ultrasound navigation based on minimally designed vehicle inspired by the bio-sonar strategy of bats, *Adv. Rob.* **33**(3–4), 169–182.
- Yin S., Liu H., Duan Z. (2021) Hourly PM2.5 concentration multi-step forecasting method based on extreme learning machine, boosting algorithm and error correction model, *Digital Signal Process.* 103221.
- Yule G. U. (1995) The foundations of econometric analysis: Why do we sometimes get nonsense correlations between time-series? *J. R. Stat. Soc.* **89**(1926), 2–9, 30–41.
- Zhao T., Zheng Y., Wu Z. (2023) Feature selection-based machine learning modeling for distributed model predictive control of nonlinear processes, *Comput. Chem. Eng.* **169**, 108074.
- Zong C., Nie J.-Y., Zhao D., Feng Y. (2015) Natural language processing and Chinese computing.

Chapter 2

Robot Navigation Position Time Series Predictive Control

2.1 Introduction

Mobile robot navigation system contains self-localization and path planning. The self-localization can obtain the current position of the robot, and path planning can generate a path to the destination. The robot navigation position modeling method can discover an underlying pattern of the robot motion, and produce a prediction position. The prediction position can be used to help robot navigation systems achieve more robust anti-collision control and health monitoring:

- (a) If the localization sensor fails, the navigation system can utilize the prediction position as the current position. The robustness of the robot navigation system can be enhanced.
- (b) Time-to-contact is an important quantitative index for robot anti-collision health state. The navigation system can compare the prediction position with the obstacle position, and estimate the time-to-contact.

Many types of research have been proposed to achieve robot position forecasting. To enhance the robustness of robot navigation when the sensor fails, (H. Liu *et al.*, 2015) adopted a hybrid position prediction method. The proposed model utilizes the Kalman filter for prediction. The initial parameters of the Kalman filter are produced by the time series model. (Sanchez-Garcia *et al.*, 2018) utilized the recursive auto-regressive exogenous model to estimate time-to-collision. The position information of the robot is generated by the monocular vision and hidden Markov chain. Several experimental studies indicate the proposed time-to-collision forecasting model can generate satisfactory forecasting results.

In this chapter, several statistical, intelligent, and deep learning models for position forecasting are introduced. The research flowchart of position series modeling is divided into four parts, including data acquisition and analysis, time series modeling, application, and conclusion. The research framework is shown in figure 2.1.

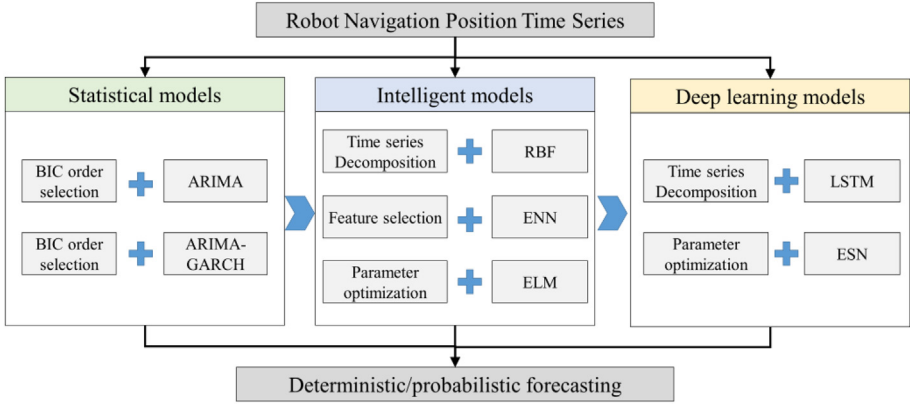


FIG. 2.1 – Research framework of chapter 1.

2.2 Robot Navigation Position Time Series Measurement

Popular self-localization methods can be divided into 6 classes by the sensors, namely mechanical methods, magnetic methods, acoustic methods, optical methods, radio methods, and other methods (Saeedvand *et al.*, 2018; Torres-Solis *et al.*, 2010). The classification is shown in figure 2.2.

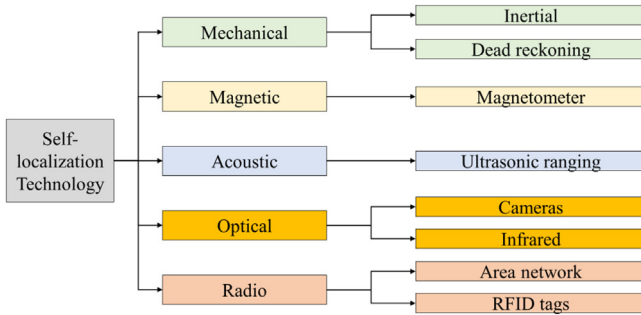


FIG. 2.2 – Classification of self-localization technology by hardware.

The mechanical methods can obtain the current position with mechanical sensors, such as an accelerometer, gyroscope, etc. Inertial and dead reckoning are two mainstream mechanical methods. The magnetic methods utilize an external magnetic sensor array for locating. The reported accuracy can reach above 99% (Son *et al.*, 2019). The acoustic methods apply the ultrasonic technique for localization. The acoustic localization methods are also widely used by animals. Bat is a prime example. Inspired by nature, several researchers proposed bioinspired robots with

acoustic localization (Yamada *et al.*, 2019; Eliakim *et al.*, 2018). Optical methods can recognize location by lights. According to the hardware, the optical methods include camera methods and infrared methods. The camera methods can obtain environmental images around the robot, and recognize neutral features or fiducial through computer vision algorithms. Robots with infrared methods can recognize the information of each infrared tag through the Infrared Data Association protocol. In the radio methods, there are two mainstream schemes, including area network methods and RFID methods. A robot in the area network methods can get its position by fingerprint technology (R. C. Luo and Hsiao, 2019). As for the RFID method, the RFID tags can emit radio signals with environmental information (DiGiampaolo and Martinelli, 2018).

All of these measurement methods can produce robot trajectory series. The introduced models in this chapter are data-driven. The historical trajectory is the only need for the data-driven models. The measurement sensor information is not necessary for the models in this chapter.

2.3 Robot Navigation Position Time Series Uncertainty Analysis

The uncertainty of the position time series is related to two aspects, including environmental uncertainty, and measurement uncertainty.

- (a) The environmental uncertainty has an impact on localization results for all sensors. For example, the camera sensors cannot detect useful features under strong lights. The failed sensors may generate invalid positions. Besides, the environmental temperature and humidity may lead to sensor drift. The sensor with drift may produce an incorrect localization position. To ensure an accurate localization position, almost all robots carry several different localization sensors against environmental uncertainty.
- (b) The measurement uncertainty contains hardware uncertainty and software uncertainty. The hardware uncertainty is caused by the inherent error of the sensor. This is determined by the sensor type. The software uncertainty is caused by the processing algorithm. For example, the computer vision processing algorithms in the optical sensor cannot generate a position without error. The software uncertainty can be alleviated by software upgrading.

2.4 Robot Navigation Position Time Series Statistical Forecasting Method

In this section, we will introduce commonly used time series forecasting models such as Autoregressive Integrated Moving Average (ARIMA) and ARIMA-Generalized Autoregressive Conditional Heteroscedasticity (GARCH) (Gheyas and Smith, 2011).

2.4.1 ARIMA Forecasting Algorithm

The ARIMA model is a typical model used in time series forecasting. There are already many forecasting models using the ARIMA method. The ARMA model can describe the auto-regressive function of a series. This characteristic is used for colored navigation noise modeling (Sheinker *et al.*, 2016; Thong-un, 2016), robot navigation control (Elnagar, 2011), etc. In this section, the ARIMA model will be applied for position-series forecasting.

(1) Modeling flow

The model flow of the ARIMA model contains three components, including preprocessing, parameter estimation, and forecasting.

- (a) In the preprocessing component, the trajectory series is divided into X-position and Y-position series. Then, the x -series and y -series are differenced. The differencing computation can make series stationary. The ARIMA can only describe the autocorrelation function of the stationary series.
- (b) In the parameter estimation component, the order of ARIMA is determined by the Bayesian Information Criterion (BIC), and the parameters of ARIMA are obtained by the maximum likelihood method.
- (c) In the forecasting component, the established model is extrapolated into the testing position time series.

A. Preprocessing

The original position time series are shown in figure 2.3. The whole dataset contains 1500 position data. The former 1000 data are used for model training, and the last 500 data are used for testing. The ARIMA model cannot forecast trajectory, so the X-position and Y-position series are used for prediction (B. Luo and Miao, 2019).

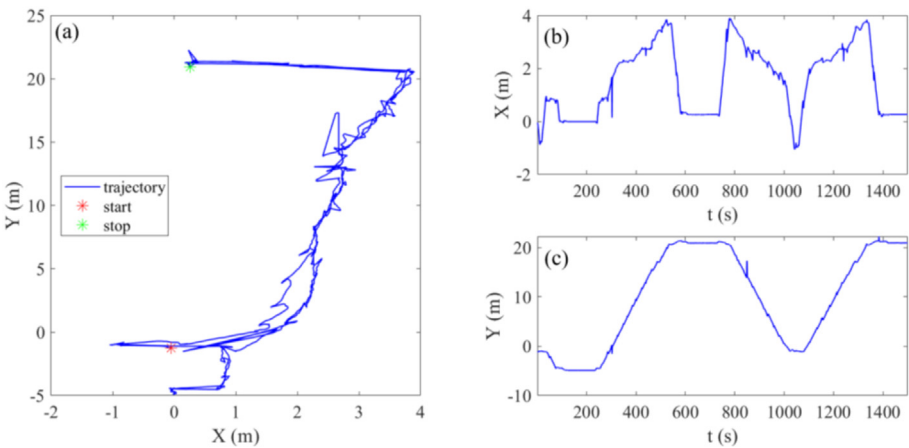


FIG. 2.3 – The original position time series for ARIMA experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

Before forecasting, the series should be transformed into a stationary series. According to the Augmented Dickey-Fuller (ADF) test results, the original X-position and Y-position series both are non-stationary (Lian *et al.*, 2020). After a one-time differencing computation, the series is stationary.

B. Parameter estimation

The form of the model is determined preliminarily according to the autocorrelation function and partial correlation function. These two functions of the differenced X-position series are shown in figure 2.4. According to figure 2.4, these two functions all have trailing features. The forecasting model should be the ARMA(p, q) model. The auto-regressive order p and moving averaging order q will be determined by the BIC criterion.

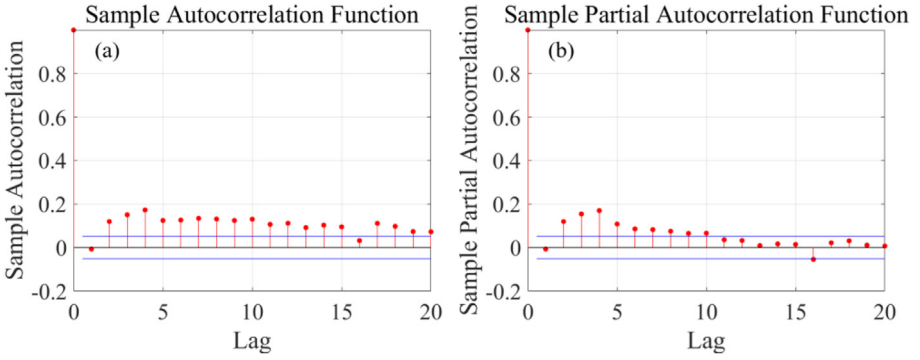


FIG. 2.4 – The autocorrelation function and partial correlation function of the differenced X-position series: (a) autocorrelation function, (b) partial correlation function.

The BIC criterion can evaluate generalized performance comprehensively. The BIC criterion contains two parts: model complexity and forecasting accuracy. The expression of the BIC criterion is presented as an equation (2.1) (Tamura *et al.*, 1991).

$$\text{BIC}(n) = \ln(n)k - 2 \ln(\hat{L}) \quad (2.1)$$

The model with minimum BIC criterion has stratified forecasting accuracy and a non-complex structure. A complicated model is more likely to obtain better forecasting accuracy in the testing set. However, the complicated model trends to over-fit and generate poor forecasting performance in the testing set. In this chapter, the grid search method is utilized to select the best model order. The auto-regressive and moving averaging orders both range within $[0, 20]$. The BIC criterion results of the X-position series can be seen in figure 2.5. It can be observed that the BIC criterion increases with the order, and the best model is ARIMA(1, 1, 2) (Contreras *et al.*, 2003).

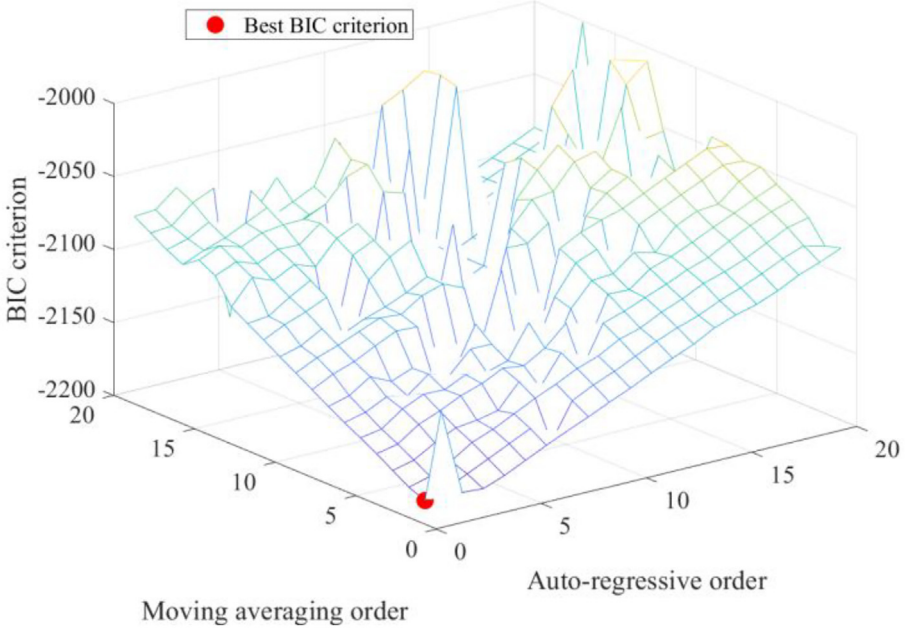


FIG. 2.5 – The BIC criterion results of the X-position series for ARIMA.

After selecting the model structure, the model parameters are estimated by the above method. The estimated ARIMA(1,1,2) is presented in equation (2.2) (Hol, 2013).

$$X_t = 0.9523X_{t-1} + \varepsilon_t - 1.1051\varepsilon_{t-1} + 0.2220\varepsilon_{t-2} \quad (2.2)$$

where X_t , X_{t-1} is the observation in time t , $t-1$. ε_t , ε_{t-1} , ε_{t-2} is the innovation in time t , $t-1$, $t-2$. The innovation ε_t obeys the Gaussian distribution $N(0, 0.0063)$.

The obtained forecasting equation is used to forecast the differenced X-position series. This equation should be transformed into equation of the original series. The ARIMA(1,1,2) should be transformed into ARMA(2,2) as equation (2.3).

$$X_t - X_{t-1} = 0.9523(X_{t-1} - X_{t-2}) + \varepsilon_t - 1.1051\varepsilon_{t-1} + 0.2220\varepsilon_{t-2} \quad (2.3)$$

So, the equation of the ARMA(2,2) is presented in equation (2.4).

$$X_t = 1.9523X_{t-1} - 0.9523X_{t-2} + \varepsilon_t - 1.1051\varepsilon_{t-1} + 0.2220\varepsilon_{t-2} \quad (2.4)$$

C. Forecasting

The forecasting result of equation (2.2) is distribution because the innovation term ε_t is a random variable. The final forecasting results are obtained by minimizing mean square error, which is presented in equation (2.5).

$$\min \left\{ E[\hat{e}_t(l)]^2 \right\} = \min \left\{ E[X_{t+l} - \hat{X}_t(l)]^2 \right\} \quad (2.5)$$

where X_{t+l} is the observation value for time $t+l$, $\hat{X}_t(l)$ is the forecasting value for time $t+l$. The minimum mean square error forecasting is presented in equation (2.6).

$$\hat{X}_t(l) = E[X_{t+l}|X_t, X_{t-1}, \dots, X_1] \quad (2.6)$$

The 1-step minimum mean square error forecasting equation can be obtained as an equation (2.7) (Lathi and Green, 2005).

$$X_t = 1.9523X_{t-1} - 0.9523X_{t-2} - 1.1051\varepsilon_{t-1} + 0.2220\varepsilon_{t-2} \quad (2.7)$$

(2) Simulative results and numerical analysis

Figure 2.6 shows the results of ARIMA prediction.

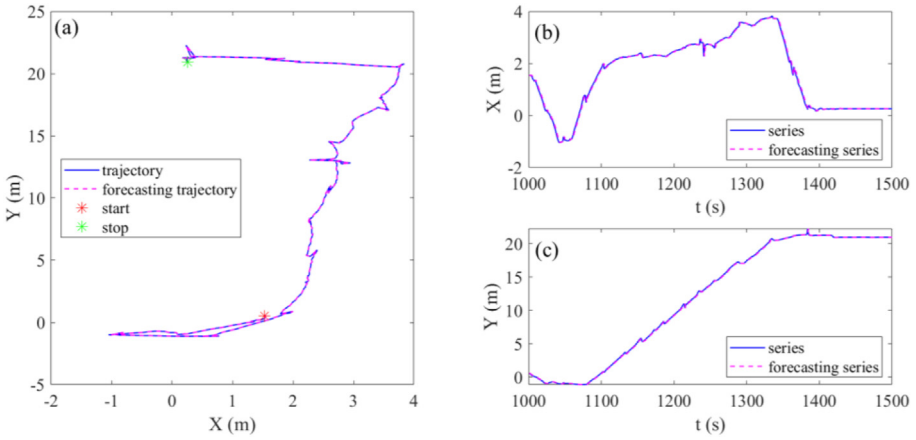


FIG. 2.6 – The forecasting results of the ARIMA model: (a) trajectory series, (b) X-position series, (c) Y-position series.

The forecasting performance can be evaluated by three performance indices, including Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) (Duan and Liu, 2019). The equations of these indices are presented as equation (2.8) (Yin *et al.*, 2021).

$$\begin{cases} \text{MAE} = \left(\sum_{t=1}^N |Y(t) - \hat{Y}(t)| \right) / N \\ \text{MAPE} = \left(\sum_{t=1}^N |(Y(t) - \hat{Y}(t)) / x(t)| \right) / N \\ \text{RMSE} = \sqrt{\left(\sum_{t=1}^N [Y(t) - \hat{Y}(t)]^2 \right) / N} \end{cases} \quad (2.8)$$

where Y is the actual value, $\hat{Y}$ is the forecasting value (Duan and Liu, 2019; K. Li *et al.*, 2017).

Table 2.1 is the evaluation index of this model.

TAB. 2.1 – Evaluation indicators of ARIMA.

	X-position series	Y-position series
MAE (m)	0.0364	0.0714
MAPE (%)	5.4939	6.1317
RMSE (m)	0.0629	0.1090

2.4.2 ARIMA-GARCH Forecasting Algorithm

ARIMA model can only realize deterministic forecasting. Probabilistic forecasting can estimate the uncertainty of the series and can obtain a prediction interval with a certain confidence level. The ARIMA-GARCH is an extension of the ARIMA model. The modeling of the GARCH is based on the result of section 2.4.1.

(1) Modeling flow

The GARCH model is built on residual. The GARCH model steps can be divided into three parts: model diagnosis, parameter estimation, and prediction (H. Liu and Duan, 2020).

In the model diagnostics component, the conditional heteroscedasticity of the forecasting residual is analyzed. The necessity to apply the GARCH model is verified.

In the parameter estimation component, the order and parameters of the GARCH are determined. The order of the model is determined by the BIC criterion, and the maximum likelihood method is used to estimate the model parameters.

In the forecasting component, the probabilistic forecasting methods for the position series and trajectory series are generated by the estimated model.

A. Model diagnostics

The X-position and Y-position residuals in section 2.4.1 contain 500 data. In this section, the first 60% of the data is the training set, and the remaining data is the test set. The trajectory, X-position, and Y-position residuals are shown in figure 2.7.

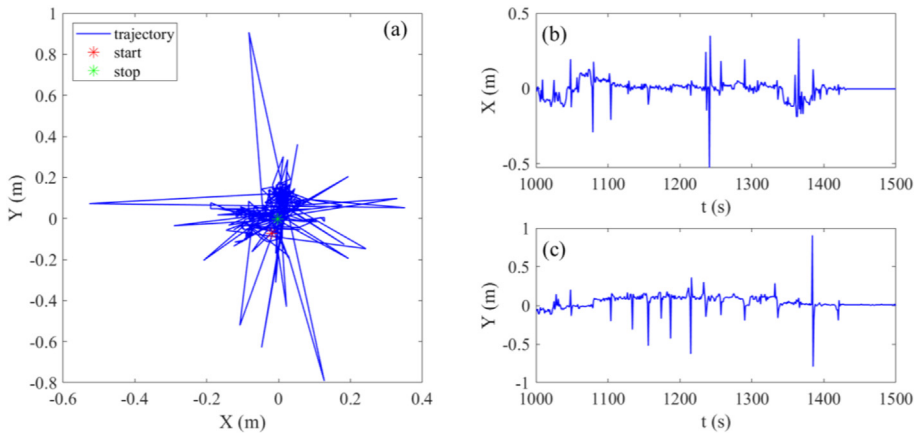


FIG. 2.7 – The forecasting residual time series: (a) trajectory series, (b) X-position series, (c) Y-position series.

The application of the GARCH needs the forecasting residual to be conditional heteroscedasticity. Before training the GARCH model, the conditional heteroscedasticity of the residual should be verified. In this chapter, the ARCH test is applied. The null hypothesis of the ARCH test is the series has no heteroscedasticity. Table 2.2 shows the test situation of ARCH.

TAB. 2.2 – ARCH test results.

	X-position series	Y-position series
p -value	2.3280E-9	0.0010

The p -values both are smaller than 1%. So the null hypothesis can be rejected significantly. This phenomenon indicates the position series have conditional heteroscedasticity.

B. Parameter estimation

The model diagnosis verifies the position series forecasting residual can be modeled by the GARCH model. The order p and q of the GARCH(p, q) model will be determined by the BIC criterion (P. Chen *et al.*, 2009). The grid research method is utilized for selecting the best parameters. The BIC criteria of different orders in the X-position series residual are shown in figure 2.8. The model with the minimum BIC criterion is GARCH(1, 5).

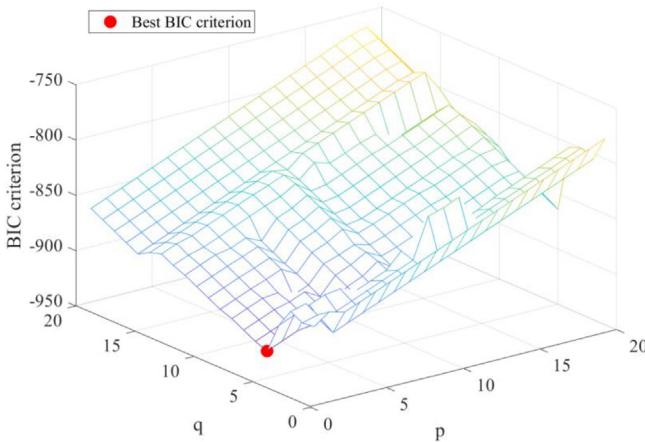


FIG. 2.8 – The BIC criterion results of the X-position series for ARIMA-GARCH.

The parameters of the GARCH model are estimated by the maximum likelihood method (Du and Swamy, 2013). The estimated model is expressed as an equation (2.9) (S. M. Liu and Brorsen, 1995).

$$\begin{cases} \varepsilon_t = \sigma_t e_t \\ \sigma_t^2 = 0.2036 + 0.1543\varepsilon_{t-1}^2 + 0.1258\varepsilon_{t-1}^2 + 0.7199\varepsilon_{t-5}^2 \end{cases} \quad (2.9)$$

where e_t obey Gaussian distribution $N(0, 1)$, ε_t is the residual at time t , σ_t^2 is the variance at time t .

C. Forecasting

Probability prediction of location is carried out through the following two steps.

- (a) The first step is to generate probabilistic predictions for the X-position and Y-position sequences.
- (b) The second step is to combine the predictions of these two positions into a trajectory probability prediction.

In the first step, the prediction intervals of the series are generated according to the forecasting results of the ARIMA and forecasting variance. The prediction interval can be presented as an equation (2.10) (Garcia *et al.*, 2005).

$$\begin{cases} \hat{Y}_{\text{lower}} = f_{\text{ARIMA}} + \alpha_{1-\frac{v}{2}}(0, \sigma^2) \\ \hat{Y}_{\text{upper}} = f_{\text{ARIMA}} + \alpha_{\frac{v}{2}}(0, \sigma^2) \end{cases} \quad (2.10)$$

where $[\hat{Y}_{\text{lower}}, \hat{Y}_{\text{upper}}]$ is the prediction interval of series, f_{ARIMA} is the forecasting results of the ARIMA model, $\alpha_{1-\frac{v}{2}}$ and $\alpha_{\frac{v}{2}}$ is the $1 - \frac{v}{2}$ and v quantile of distribution $N(0, \sigma)$, σ^2 is the forecasting variance of the GARCH model (H. Liu, 2020). In this chapter, the v is 0.99, *i.e.*, 99% actual value will be included in the prediction interval.

In the second step, the prediction interval of the trajectory is generated with a specific shape. Several shapes of the trajectory prediction interval are shown in figure 2.9.

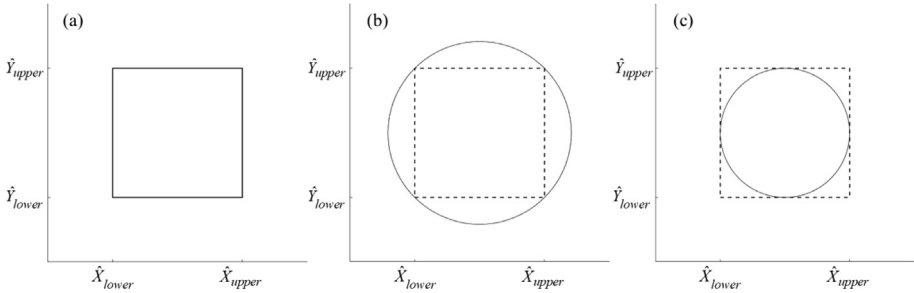


FIG. 2.9 – The different shapes of the trajectory prediction interval: (a) rectangle, (b) excircle, (c) incircle.

The best shape of the trajectory prediction is selected with the historical data. The scatters forecasting residuals between the X-position and Y-position series are shown in figure 2.10. To explain the relationship between the residual values and

prediction intervals, the prediction intervals are normalized into $[-1,1]$, and the residual values are normalized as an equation (2.11) (Teal *et al.*, 2006). The normalization ratios of the prediction interval and residual are identical.

$$Y'_r = \frac{Y_r}{\hat{Y}_{\text{upper}} - f_{\text{ARIMA}}} \quad (2.11)$$

where Y'_r is the normalized forecasting residual, and Y_r is the forecasting residual.

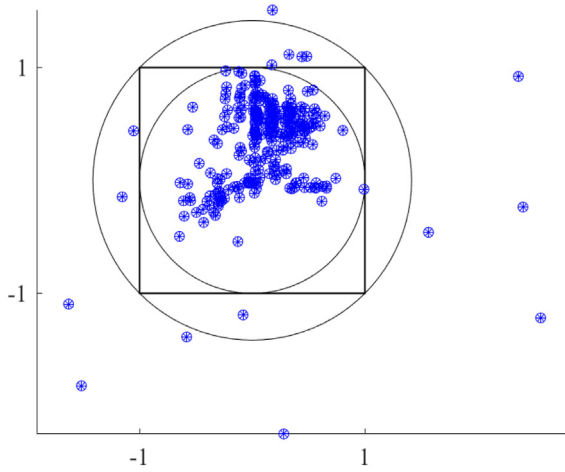


FIG. 2.10 – The scatters of forecasting residuals between the X-position and Y-position series.

The cover percentages of each shape are presented in table 2.3. The cover percentages of the rectangle and incircle are identical. However, the cover percentage is 94.48%. This cover percentage is far lower than the one of the excircle. So, the excircle is selected as the final shape.

TAB. 2.3 – Cover percentages of each shape.

	Rectangle	Excircle	Incircle
Cover percentage	94.48%	96.90%	94.48%

(2) Simulative results and numerical analysis

Figure 2.11 shows the probability prediction results.

The probabilistic prediction performance evaluation expressions are given as equation (2.12) (Khosravi *et al.*, 2013). Among them, the Prediction Interval Coverage Probability (PICP) represents the coverage probability of the prediction interval, the Coverage Width-based Criterion (CWC) represents the combined standard based on the coverage width, and the Prediction Interval Normalized Average Width (PINAW) represents the normalized average width of the prediction interval.

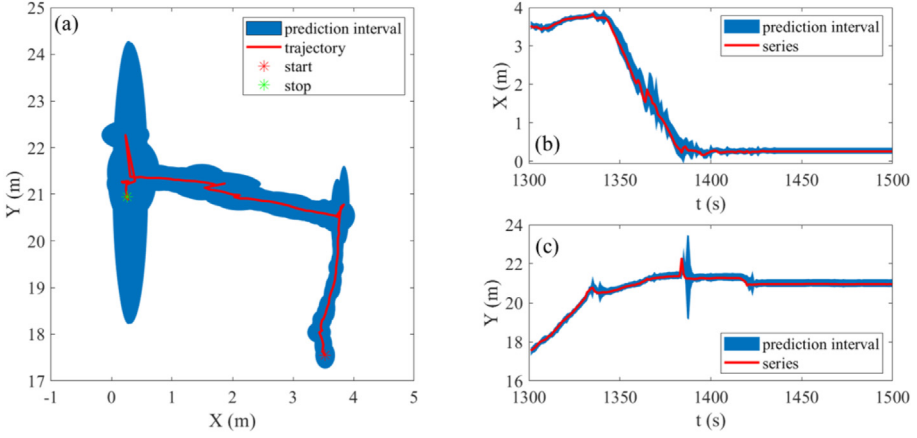


FIG. 2.11 – The probabilistic forecasting results of the ARIMA-GARCH: (a) trajectory series, (b) X-position series, (c) Y-position series.

$$\left\{ \begin{array}{l} \text{PICP} = \frac{1}{n} \sum_{t=1}^n c_t \\ \text{PINAW} = \frac{1}{nR} \sum_{t=1}^n (U_t - L_t) \\ \text{CWC} = \text{PINAW} (1 + \xi(\text{PICP}) e^{-\eta(\text{PICP}-v)}) \end{array} \right. \quad (2.12)$$

where U_t and L_t are the upper and lower bounds of the prediction interval (Jia *et al.*, 2021). If the observation is included by the prediction interval, *i.e.*, $x_t \in [L_t, U_t]$, then $c_t = 1$, otherwise $c_t = 0$. If $\text{PICP} \geq v$, $\xi(\text{PICP}) = 0$, otherwise $\xi(\text{PICP}) = 1$. R is equal to the range of the actual series. The confidence level v is equal to 99%, and the trade-off parameter η is equal to 50.

Equation (2.12) is applied to evaluate the X-position and Y-position series probabilistic forecasting. The evaluation of the trajectory series probabilistic forecasting has a slight difference from equation (2.12). The calculation of the PINAW is modified as an equation (2.13) (H. Liu, Duan *et al.*, 2019).

$$\text{PINAW} = \frac{1}{nR_x R_y} \sum_{t=1}^n S_t \quad (2.13)$$

where S_t is the area of the trajectory prediction interval, R_x and R_y are the range of the X-position and Y-position series.

The probabilistic forecasting evaluation indices are presented in table 2.4.

TAB. 2.4 – The probabilistic evaluation indices of the ARIMA-GARCH model.

	X-position series	Y-position series	Trajectory
PICP	98.50%	98.00%	98.50%
PINAW	0.0736	0.0962	0.0113
CWC	0.1682	0.2548	0.0259

2.5 Robot Navigation Position Time Series Intelligent Forecasting Method

The above statistical methods can only express the linear information of the position series (Duan and Liu, 2019). This section introduces several intelligent methods that can predict the nonlinear components of the position sequence, such as extreme learning machines, Radial Basis Function (RBF) neural networks, and Elman neural networks. Several methods are used to enhance forecasting performance, including decomposition, feature selection, and optimization methods.

2.5.1 RBF Neural Network Forecasting Algorithm

The RBF neural network is a classical single hidden layer feed-forward neural network (C. Li and Tang, 2016). To improve the predictive performance of the model, the decomposition algorithm is applied. The Wavelet Packet Decomposition (WPD) algorithm can separate the series into more predictable components (H. Liu, 2020).

(1) Modeling flow

The modeling flow of the RBF neural network with the decomposition algorithm contains four components, including preprocessing, decomposition, training, and forecasting (H. Liu, Xu *et al.*, 2019).

- (a) In the preprocessing component, the position trajectory series is divided into the X-position and Y-position series.
- (b) In the decomposition component, the X-position and Y-position series are decomposed into several subseries by the WPD.
- (c) In the training component, the RBF models are trained with each subseries.
- (d) In the forecasting component, the forecasting results in each subseries are summed up into the final series.

A. Preprocessing

The modeling data contains 1500 data, the 1st–1000th is training data for training the RBF model, and the 1001st–1500th is testing data for predictive performance evaluation. Figure 2.12 shows the original trajectory and X-position and Y-position series.

B. Decomposition

The X-position and Y-position series are decomposed by the WPD. The obtained subseries have a narrower frequency band than the original series. Taking the X-position series as an example, the subseries and their frequency spectrum are shown in figure 2.13. Each row represents the time and frequency domain of a subseries. The wavelet is ‘db5’, and the decomposition level is 2. It can be observed that the frequency components of each series are different. The subseries represent different oscillate characteristics in multi-resolution.

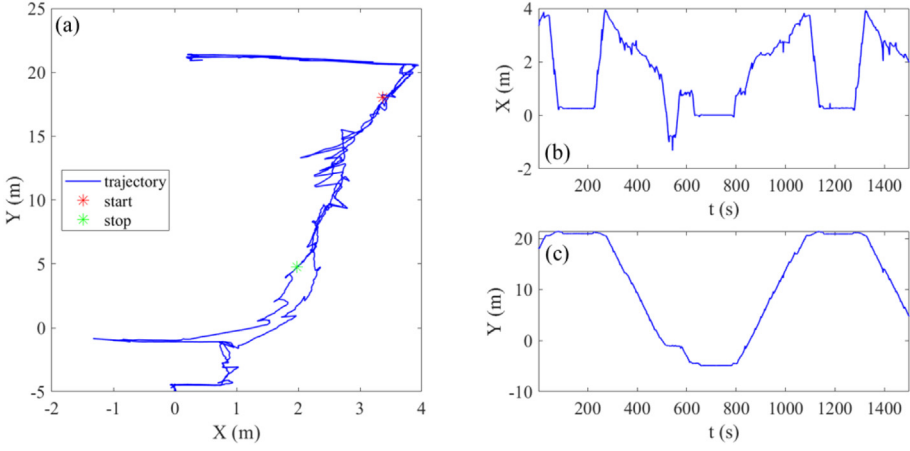


FIG. 2.12 – The original position time series for RBF neural network experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

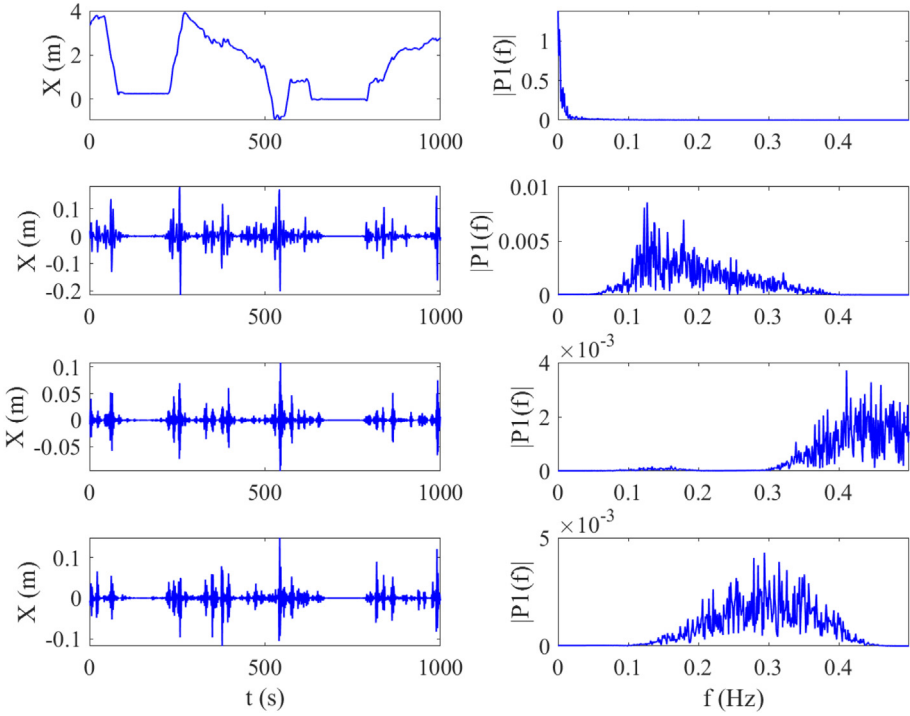


FIG. 2.13 – The subseries and their frequency spectrum of the X-position series.

C. Training

After decomposition, the RBF model is trained by each subseries. RBF is the active function of the hidden layer (Ding *et al.*, 2011). Set the maximum number of RBF training iterations to 200. The spread of the radial basis functions is set to 10. The loss curves during the training of the X-position series are shown in figure 2.14.

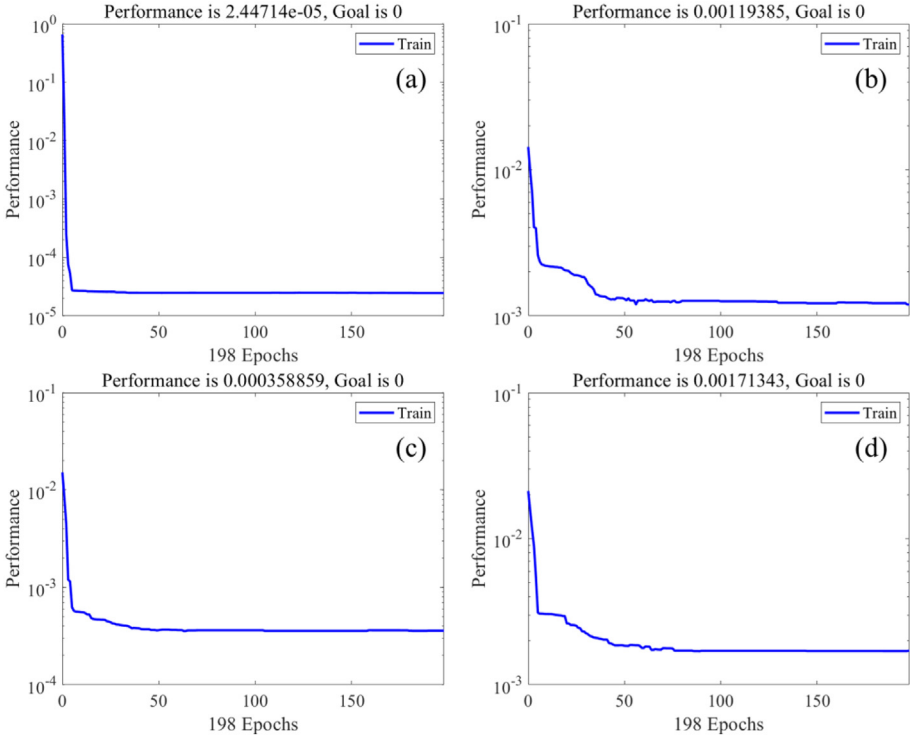


FIG. 2.14 – The loss during the training of the RBF neural network in the X-position series: (a) subseries #1, (b) subseries #2, (c) subseries #3, (d) subseries #4.

D. Forecasting

The forecasting results in each subseries are summed up to the final forecasting results.

(2) Simulative results and numerical analysis

The forecasting results of the WPD-RBF model and RBF models are shown in figure 2.15.

The evaluation indices of these two models are presented in table 2.5. The subseries of the WPD model has simpler frequency components than the original series. The RBF model can predict subseries accurately. Therefore, the WPD-RBF is more predictable.

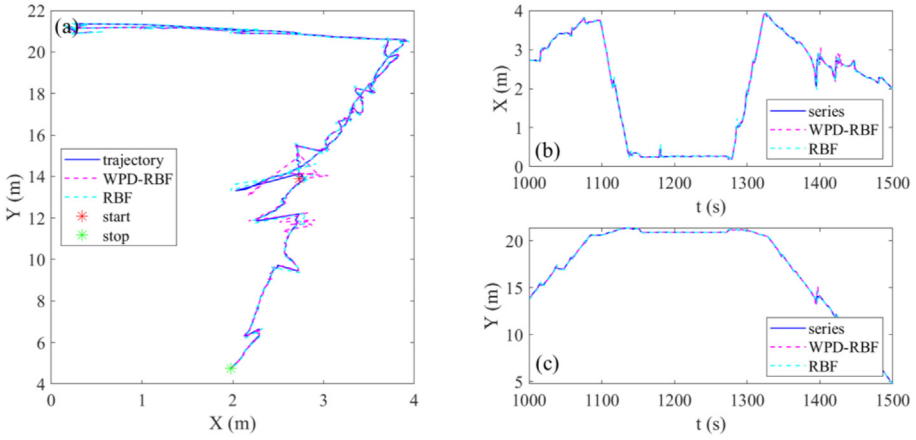


FIG. 2.15 – The forecasting results of the WPD-RBF and RBF models: (a) trajectory series, (b) X-position series, (c) Y-position series.

TAB. 2.5 – The evaluation indices of the WPD-RBF and RBF models.

Series	Evaluation indices		
	MAE (m)	MAPE (%)	RMSE (m)
WPD-RBF			
X-position series	0.0129	0.0079	0.0295
Y-position series	0.0291	2.6426	0.0649
RBF			
X-position series	0.0196	0.1460	0.0687
Y-position series	0.0451	0.3136	0.0885

2.5.2 Elman Neural Network Forecasting Algorithm

Elman Neural Network (ENN) is a recurrent network (Ren *et al.*, 2009). In this section, the ENN is combined with the Binary Genetic Algorithm (BGA). The BGA algorithm can select the input structure, and alleviate the curse of dimensionality.

(1) Modeling flow

The modeling flow of the Elman neural network model contains four components, including preprocessing, feature selection, training, and forecasting.

- In the preprocessing component, the position trajectory is divided into the X-position and Y-position series.
- In the feature selection component, the input structure of the ENN is selected by the BGA algorithm.
- In the training component, the input of the ENN is the selected feature, and the output is the predicted value.

- (d) In the forecasting component, the trained ENN is used to generate forecasting values.

A. Preprocessing

The position trajectory series is separated into the X-position and Y-position series. The original trajectory and X-position and Y-position series are shown in figure 2.16. The whole data set contains 1500 data. The former 1000 data are applied for training, the remaining data is used for testing.

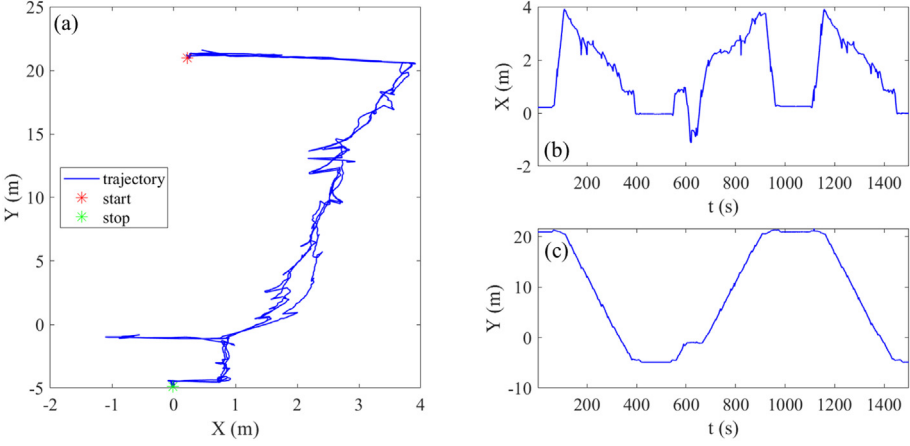


FIG. 2.16 – The original position time series for Elman neural network experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

B. Feature selection

The BGA algorithm is utilized for feature selection. The features for prediction Y_t are the historical data $[Y_{t-N+1}, \dots, Y_{t-1}]$, where N is the number of the features (in this chapter, the N is 5). The optimization function of the BGA algorithm is minimizing mean square error, which is presented as an equation (2.14) (Onieva *et al.*, 2015).

$$\min_w \sum_t \frac{(\hat{Y}_t(w) - Y)^2}{T} \quad (2.14)$$

where w consists of 0 and 1. The number of the w is equal to the number of the features (Schalkoff, 2007). If the i -th element of w is equal to 1, then the i -th feature is applied for prediction. $\hat{Y}(w)$ is the forecasting series generated with features selected according to w .

The number of features after BGA optimization is reduced, and the optimization results are shown in table 2.6.

TAB. 2.6 – The feature selection results of the BGA.

	Y_{t-5}	Y_{t-4}	Y_{t-3}	Y_{t-2}	Y_{t-1}
X-position series	1	0	0	0	1
Y-position series	0	0	0	1	1

C. Training

The training of the Elman neural work is carried out with a gradient descent algorithm. The number of hidden units is 11. The loss during training is shown in figure 2.17. The number of training iterations in the Y-position series is less than the one in the X-position series. This phenomenon is due to the early stop mechanism. This mechanism can prevent over-fitting.

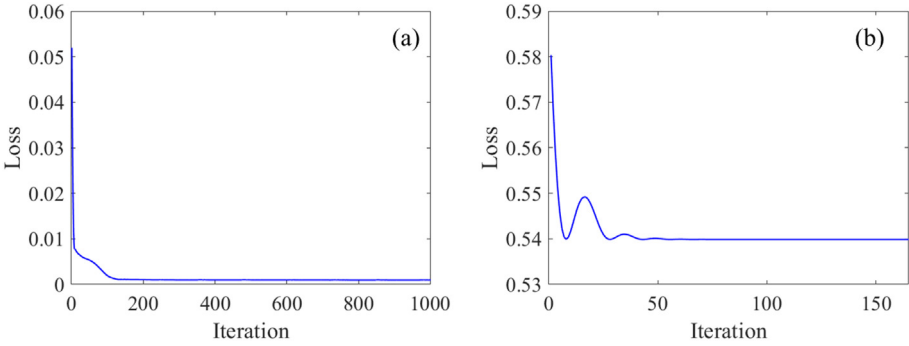


FIG. 2.17 – The loss during the training of the Elman neural network: (a) X-position series, (b) Y-position series.

D. Forecasting

The trained ENN is utilized for forecasting testing series of the X-position and Y-position series.

(2) Simulative results and numerical analysis

The forecasting results of the BGA-Elman and ENN models are shown in figure 2.18.

The evaluation indices of these models are presented in table 2.7. The BGA-Elman outperforms Elman significantly. The BGA can select the optimized features for the Elman neural network. The selected features are a subset of the original features. This subset contains enough prediction information and alleviates the curse of the dimension. So, the BGA-Elman can achieve better results than the Elman neural network model.

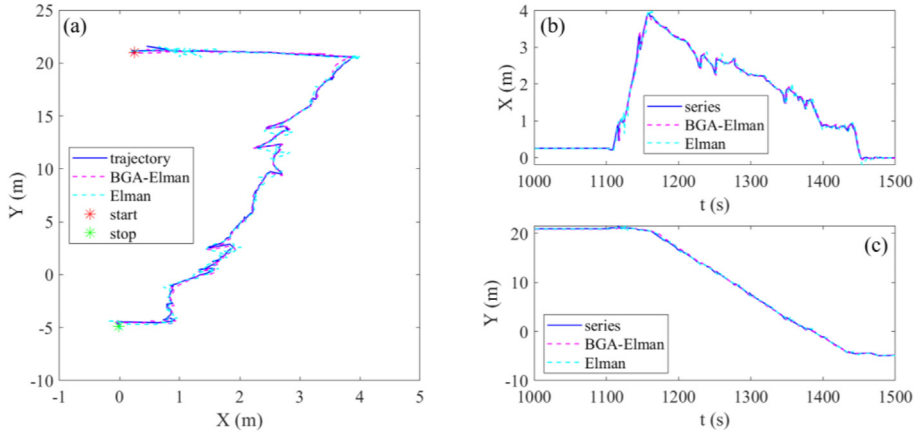


FIG. 2.18 – The forecasting results of the BGA-Elman and Elman models: (a) trajectory series, (b) X-position series, (c) Y-position series.

TAB. 2.7 – The evaluation indices of the BGA-Elman and Elman models.

Series	Evaluation indices		
	MAE (m)	MAPE (%)	RMSE (m)
BGA-Elman			
X-position series	0.0316	11.7959	0.0637
Y-position series	0.1127	3.1041	0.1502
Elman			
X-position series	0.0498	21.7040	0.0872
Y-position series	0.1257	7.1487	0.1823

2.5.3 Extreme Learning Machine Forecasting Algorithm

The Extreme Learning Machine (ELM) model is a variant of a single hidden layer feed-forward neural network (K. Li *et al.*, 2017; C. Li and Tang, 2016). The training of the ELM is utilized by the matrix calculation. The ELM is applied to estimate robot displacement in the robot navigation system (Moskalenko *et al.*, 2019). The performance of the ELM model is affected greatly by the initial weights and bias. In this chapter, the initial weights and biases are optimized by the Grey Wolf Optimization (GWO) algorithm.

(1) Modeling flow

The modeling flow of the ELM model includes four components: preprocessing, optimization, training, and forecasting.

- (a) In the preprocessing component, the position trajectory is divided into the X-position and y-position series.

- (b) In the optimization component, the initial weights and bias of the ELM model are optimized by the GWO model.
- (c) In the training component, the ELM models are trained. The initial weights and bias are the optimized ones.
- (d) In the forecasting component, the trained ELM is applied to forecast the X-position and y-position series.

A. Preprocessing

The original trajectory and X-position and Y-position series are shown in figure 2.19. The whole data set contains 1500 data. The former 1000 data are applied for optimization and training, and the last 500 data are applied for forecasting.

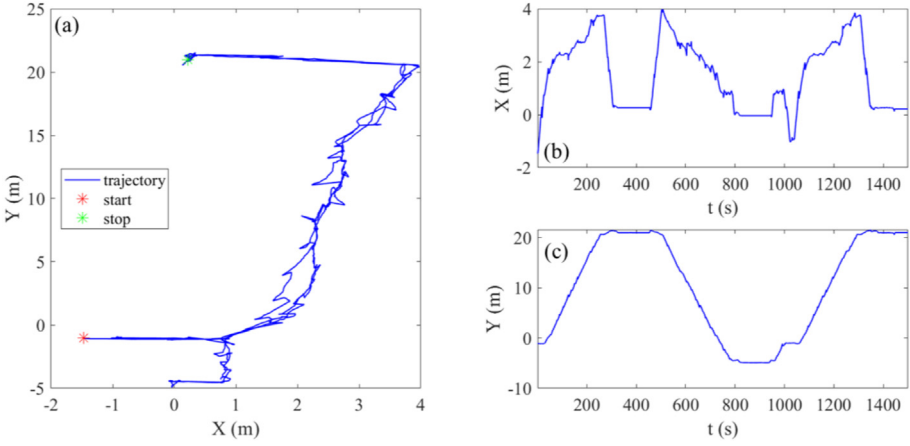


FIG. 2.19 – The original position time series for extreme learning machine experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

B. Optimization

The equation of the ELM is presented as equation (2.15) (Alom *et al.*, 2019).

$$\sum_{l=1}^L \beta_l g(\mathbf{w}_l \cdot \mathbf{X}_n + b_l) = Y_n \quad n = 1, 2, \dots, N \quad (2.15)$$

where $\mathbf{X}$ represents input, Y represents output, $\mathbf{w}$ represents weights, b represents bias, $g(\bullet)$ is the active function, and β is weights connecting the output layer. Only the output weights β will be trained during the training of the ELM, the weights $\mathbf{w}$, and bias b will not be trained (Alom *et al.*, 2019).

The initial weights $\mathbf{w}$ and bias b of the ELM model are optimized by the GWO model. The objective function of the GWO model is minimizing mean square error, which is presented as an equation (2.16) (Ganjidoost *et al.*, 2016).

$$\min_w \sum_t \frac{(\hat{Y}_t(\mathbf{w}, b) - Y)^2}{T} \quad (2.16)$$

where $\hat{Y}(\mathbf{w}, b)$ is the forecasting results with initial weights $\mathbf{w}$ and bias b . Taking the Y-position series as an example, the loss during the training processing is shown in figure 2.20. It can be observed that the optimization of the GWO converges.

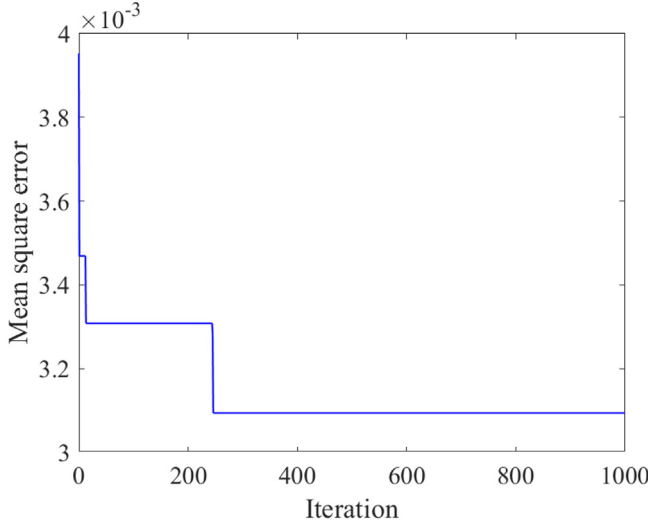


FIG. 2.20 – The loss during the optimization of the GWO in Y-position series.

C Training

The training processing of the ELM is carried out by matrix computation. Equation (2.15) can be explained as equation (2.17) (Man *et al.*, 2011).

$$\mathbf{H}\beta = \mathbf{y} \quad (2.17)$$

where

$$\mathbf{H}(\mathbf{w}_1, \dots, \mathbf{w}_L, \mathbf{x}_1, \dots, \mathbf{x}_L, b_1, \dots, b_L) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_1 + b_L) \\ \vdots & \dots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \dots & g(\mathbf{w}_L \cdot \mathbf{x}_N + b_L) \end{bmatrix}_{N \times L},$$

$\beta = [\beta_1, \beta_2, \dots, \beta_L]^T$, $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$, $\mathbf{w}$ and b are the optimized weights and bias. The β can be solved as equation (2.18).

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{y} \quad (2.18)$$

where $\mathbf{H}^\dagger$ is the pseudo-inverse matrix of $\mathbf{H}$.

D. Forecasting

The trained ELM model is applied to generate forecasting value in testing data.

(2) Simulative results and numerical analysis

The forecasting results of the GWO-ELM and ELM are shown in figure 2.21.

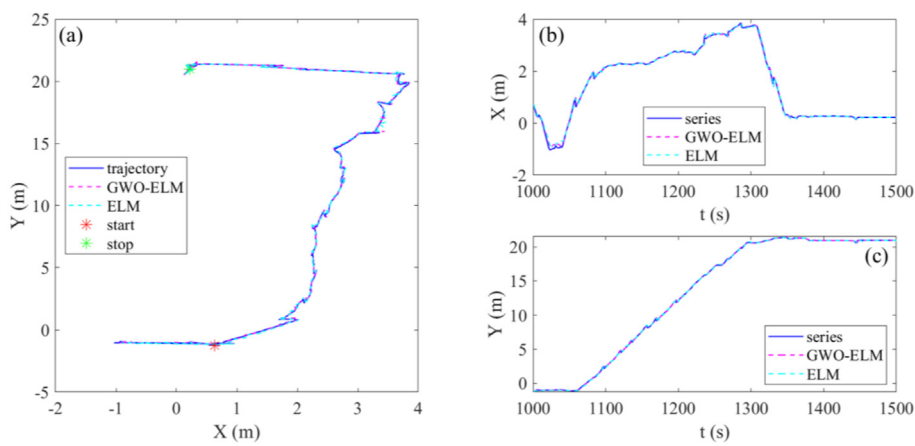


FIG. 2.21 – The forecasting results of the GWO-ELM and ELM model: (a) trajectory series, (b) X-position series, (c) Y-position series.

Table 2.8 is the evaluation index. The GWO-ELM model outperforms the ELM model significantly. The ELM model sets initial weights and bias with random variables. The trained ELM may be only locally optimal. The GWO model can search the global optimized weights and bias. With these weights and biases, the ELM model can achieve global optimization. So the forecasting performance of the ELM model can be enhanced.

TAB. 2.8 – The evaluation indices of the GWO-ELM and ELM models.

Series	Evaluation indices		
	MAE (m)	MAPE (%)	RMSE (m)
GWO-ELM			
X-position series	0.0270	3.8939	0.0515
Y-position series	0.0373	1.4567	0.0720
ELM			
X-position series	0.0336	4.7388	0.0617
Y-position series	0.0634	3.1981	0.0968

2.6 Robot Navigation Position Time Series Deep Learning Forecasting Method

The deep learning models have better nonlinear fitting performance than the classical intelligent models (Zong *et al.*, 2015). This section introduced three different deep learning methods for position forecasting, including Long Short-Term Memory (LSTM) and Echo State Network (ESN). The decomposition and optimization algorithms are applied to enhance forecasting performance.

2.6.1 LSTM Deep Neural Network Forecasting Algorithm

The LSTM model is a kind of recurrent neural network (Jiuping *et al.*). This model can address gradient exploration and vanishing problems during training. The LSTM model is applied in the navigation with spatial and temporal features, due to the powerful fitting capacity of the LSTM model (Kostavelis *et al.*, 2016). In this section, the LSTM model is combined with the Singular Spectrum Analysis (SSA) algorithm. The SSA model can decompose the original series into subseries. The LSTMs are applied to forecast all subseries.

(1) Modeling flow

The modeling flow of the LSTM model contains four components, including pre-processing, decomposition, training, and forecasting (Xydas *et al.*, 2017).

- (a) In the pre-processing component, the position trajectory is divided into the X-position and Y-position series.
- (b) In the decomposition component, the X-position and Y-position series are decomposed into subseries.
- (c) In the training component, The LSTM is built to fit each subseries.
- (d) In the forecasting component, the trained LSTM model is utilized for forecasting.

A. Preprocessing

The original position trajectory contains an X-position and Y-position series. The original trajectory and X-position and Y-position series are shown in figure 2.22. There are 1500 data in the dataset. The 1st–1000th data are applied for training, and the 1001st–1500th data are applied for forecasting.

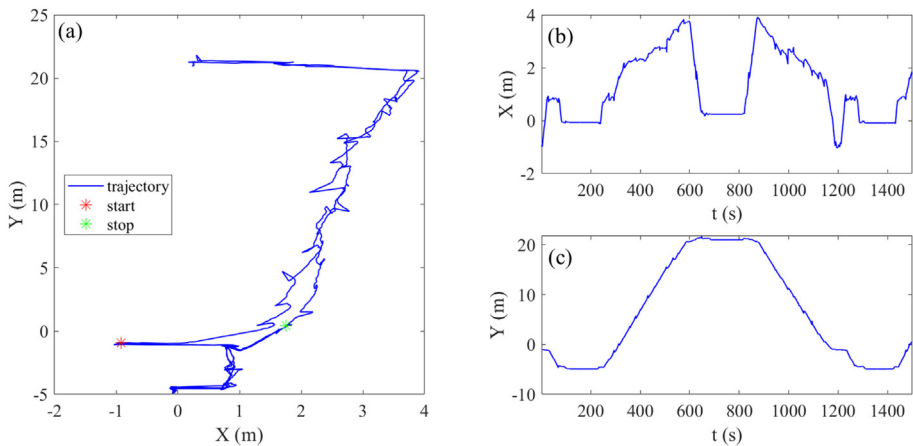


FIG. 2.22 – The original position time series for LSTM experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

B. Decomposition

Figure 2.23 shows the time domain and frequency domain of each subsequence in the X-position series. Each row represents the time and frequency domain of a subseries. The number of decomposition subseries is set to 4. It can be observed that the frequency components of each subseries are different, and modal aliasing is less.

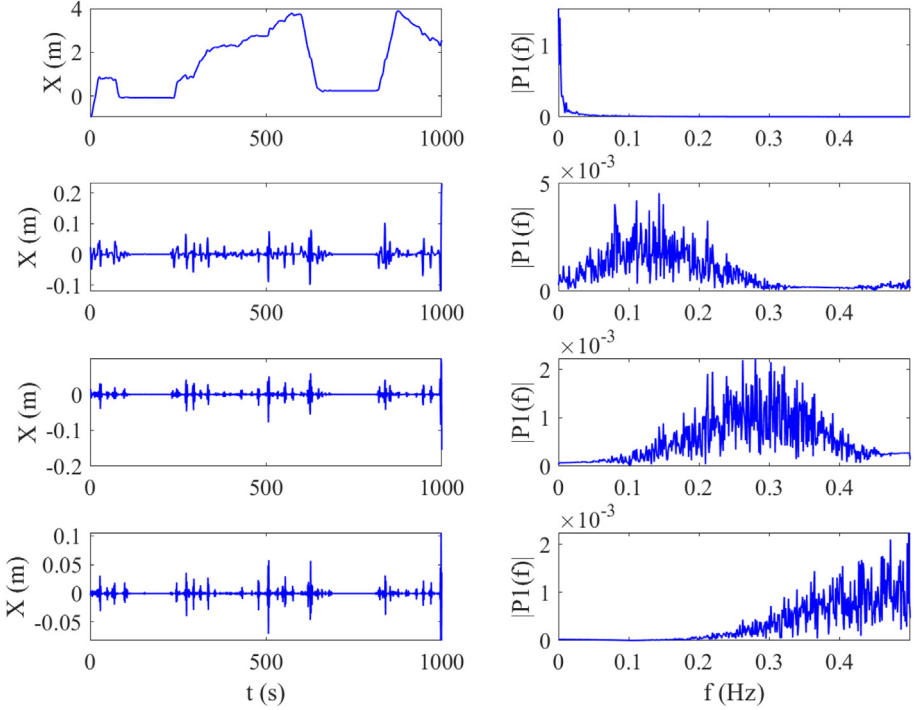


FIG. 2.23 – The subseries and their frequency spectrum of the X-position series.

C. Training

The LSTM model is trained by each subseries. The training optimizer of LSTM is the RMS Prop algorithm, the L2 regularization rate is 0.05, the maximum epoch is 250, and there are 300 hidden units. The loss curves of the X-position subseries during training are presented in figure 2.24. These curves are processed by average filtering.

D. Forecasting

The trained LSTM models are used to generate forecasting results in all subseries, and all forecasting results are summed up to the final forecasting results.

(2) Simulative results and numerical analysis

Figure 2.25 shows the forecasting results.

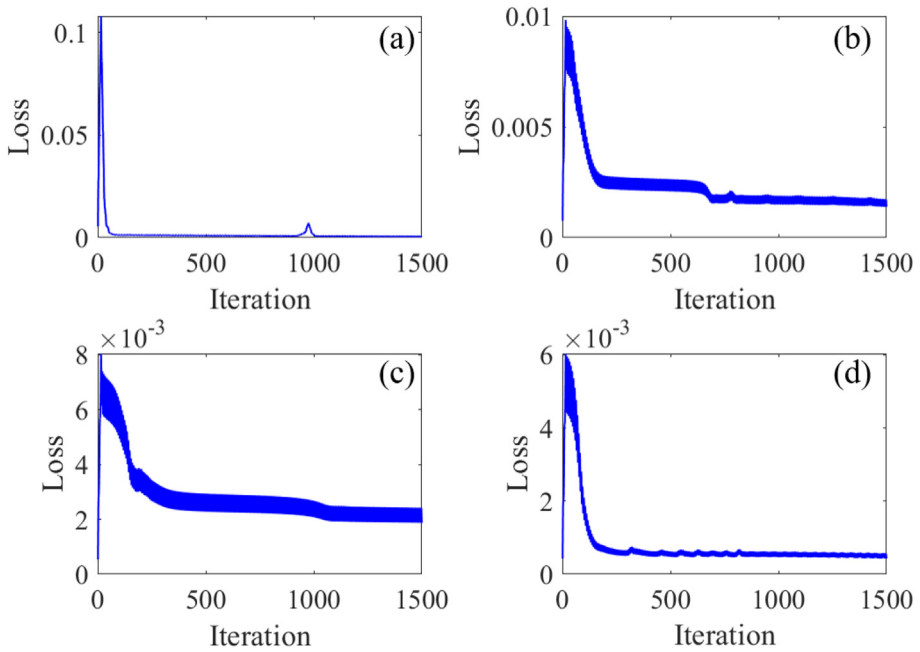


FIG. 2.24 – The Loss during the training of the LSTM: (a) subseries #1, (b) subseries #2, (c) subseries #3, (d) subseries #4.

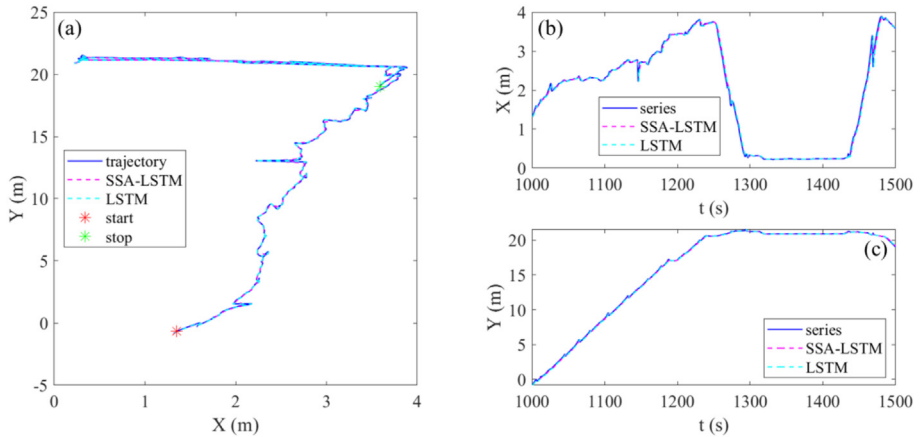


FIG. 2.25 – The forecasting results of the SSA-LSTM and LSTM models: (a) trajectory series, (b) X-position series, (c) Y-position series.

The evaluation indices of these two models are presented in table 2.9. The SSA-LSTM model is better than the LSTM model. The subseries obtained by the SSA have a simple oscillating mode. The LSTM can forecast the subseries easily. So the SSA-LSTM is better.

TAB. 2.9 – The evaluation indices of the SSA-LSTM and LSTM models.

Series	Evaluation indices		
	MAE (m)	MAPE (%)	RMSE (m)
SSA-LSTM			
X-position series	0.0297	2.1621	0.0531
Y-position series	0.0663	4.6808	0.0858
LSTM			
X-position series	0.0339	2.3782	0.0709
Y-position series	0.0749	4.6420	0.0999

2.6.2 ESN Deep Neural Network Forecasting Algorithm

The ESN model is a reservoir computing model. The ESN model is a special recurrent neural network where only the output weights are trained. Other weights are initialized randomly. In this section, the Particle Swarm Optimization (PSO) algorithm is used to optimize the weights to improve the prediction effect.

(1) Modeling flow

The modeling flow of the ESN model includes four components: preprocessing, optimization, training, and forecasting.

- In the preprocessing component, the position trajectory is divided into the X-position and Y-position series.
- In the optimization component, the weight of the ESN model is generated by the PSO algorithm.
- In the training component, the ESN model is trained by matrix computation with the optimized weights.
- In the forecasting component, the trained ESN is used for prediction.

A. Preprocessing

The position trajectory series is divided into the X-position and Y-position series. The ESN model is built on each position series. The original trajectory and X-position and Y-position series are shown in figure 2.26. The 1st–1000th data are used for optimizing and training the ESN, and the 1001st–1500th data are used for testing the forecasting performance.

B. Optimization

The structure of the ESN is shown in figure 2.27.

Let input weights, reservoir, feedback, and output weights be W_{in} , W , W_{back} , and W_{out} . The equation between input X and output Y at time t is presented as equations (2.19) and (2.20) (Dorffner, 2001).

$$u(t) = f(W_{\text{in}} \times X(t) + W \times u(t-1) + W_{\text{back}} \times Y(t-1)) \quad (2.19)$$

$$Y(t) = g(W_{\text{out}}[u(t); X(t)]) \quad (2.20)$$

where $u(t)$ are states of the reservoir layer at time t , $f(\bullet)$, $g(\bullet)$ are active functions of reservoir and output layers.

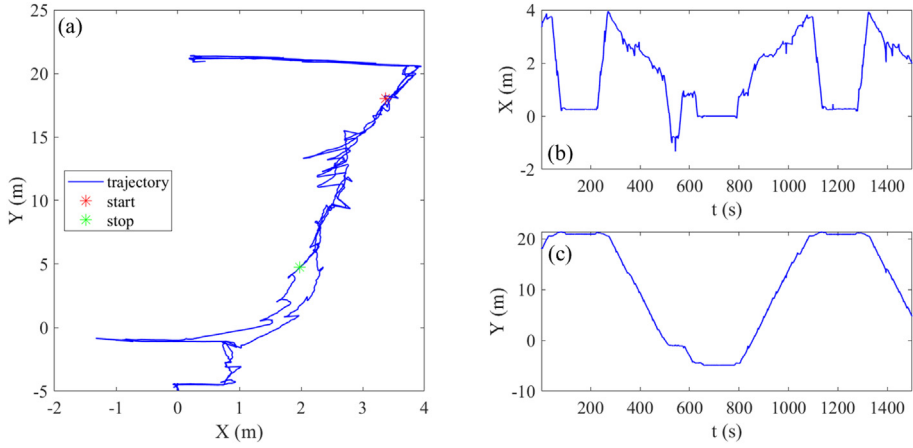


FIG. 2.26 – The original position time series for ESN experiments: (a) trajectory series, (b) X-position series, (c) Y-position series.

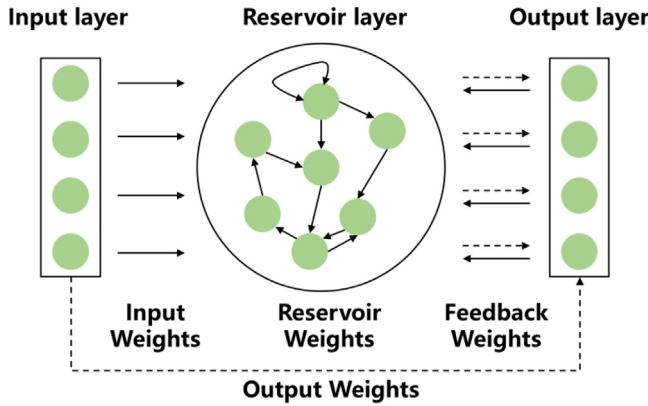


FIG. 2.27 – The structure of the ESN model.

The input, reservoir, and feedback weights are initialized, and will not be trained. These weights $\mathbf{w}$ of the ESN model are optimized by the PSO algorithm. In this section, the number of the input, reservoir, and output units are 5, 5, and 1, respectively. The objective function of the PSO is minimizing mean square error, which is presented as an equation (2.21) (Ganjidoost *et al.*, 2016).

$$\min_w \sum_t \frac{(\hat{Y}_t(\mathbf{w}) - Y)^2}{T} \quad (2.21)$$

where $\hat{Y}(\mathbf{w})$ is the forecasting results with initial weights $\mathbf{w}$. Taking the X-position series as an example, the loss during the optimization processing is shown in figure 2.28. It can be observed that the PSO algorithm converges.

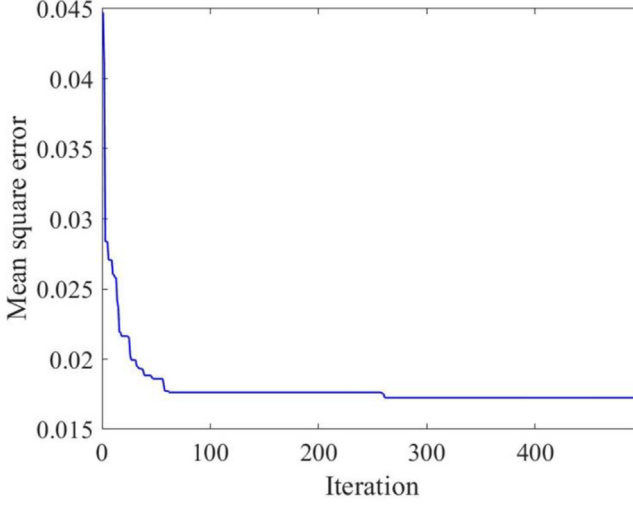


FIG. 2.28 – The Loss during the optimization of the PSO in X-position series.

C. Training

The training processing of the ESN is calculated by matrix computation. At first, the first input sample $X(1)$ is fed into equations (2.22) and (2.23) (Lin *et al.*, 2009).

$$u(1) = f(W_{\text{in}} \times X(1) + W \times u(0) + W_{\text{back}} \times Y(0)) \quad (2.22)$$

$$Y(1) = g(W_{\text{out}}[u(1); X(1)]) \quad (2.23)$$

where $u(0) = Y(0) = 0$.

Then other input samples are fed into equation (2.19) to obtain output $Y(2), Y(3), \dots, Y(N)$, where N is the number of the input samples (Cho *et al.*, 2018).

The trained output matrix can be obtained by equation (2.24).

$$W_{\text{out}} = (M^{-1} \times T)^T \quad (2.24)$$

where $M = [X(1), X(1), \dots, X(N)]$, $T = [Y(1), Y(1), \dots, Y(N)]^T$

D. Forecasting

The trained ELM model is applied to generate forecasting value in testing data.

(2) Simulative results and numerical analysis

The forecasting results of the PSO-ESN and ESN are shown in figure 2.29.

The evaluation indices are presented in table 2.10. The PSO-ESN model is better than the ESN model. With the random initial weights, the trained output weights of the ESN may be locally optimal for prediction. The PSO algorithm can generate globally optimized weights, and the ESN can generate globally optimized output weights. So the PSO can be helpful to enhance the forecasting performance.

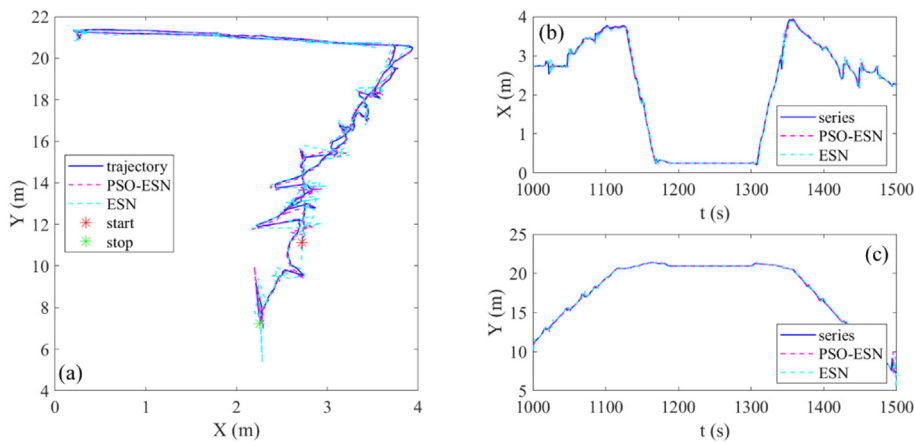


FIG. 2.29 – The forecasting results of the PSO-ESN and ESN model: (a) trajectory series, (b) X-position series, (c) Y-position series.

TAB. 2.10 – The evaluation indices of the PSO-ESN and ESN models.

Series	Evaluation indices		
	MAE (m)	MAPE (%)	RMSE (m)
PSO-ESN			
X-position series	0.0330	2.1142	0.0631
Y-position series	0.0534	0.4131	0.1507
ESN			
X-position series	0.0366	2.3946	0.0691
Y-position series	0.0971	0.7331	0.2073

2.7 Comparative Analysis of Forecasting Performance

Sections 2.4–2.6 introduce several different forecasting models. The forecasting models are compared with 10 different position series comprehensively. Six models are utilized in forecasting performance comparison, including the ARIMA, WPD-RBF, BGA-Elman, GWO-ELM, SSA-LSTM, and PSO-ESN models. The forecasting performance ranks are shown in figure 2.30. Rank 1 means the model is the best in the series, rank 6 means the model is the worst. From figure 2.30, it can be observed that the ARIMA and WPD-RBF model are the best models. Almost all rank 1 is achieved by these two models. The deep learning models perform worse. These two models have only one-time rank 1 in series #7. Almost all ranks of the PSO-ESN model are 5. This phenomenon may be caused by overfitting. The deep learning models are too complicated, so the generation performance is worse than the simple statistical and intelligent models.

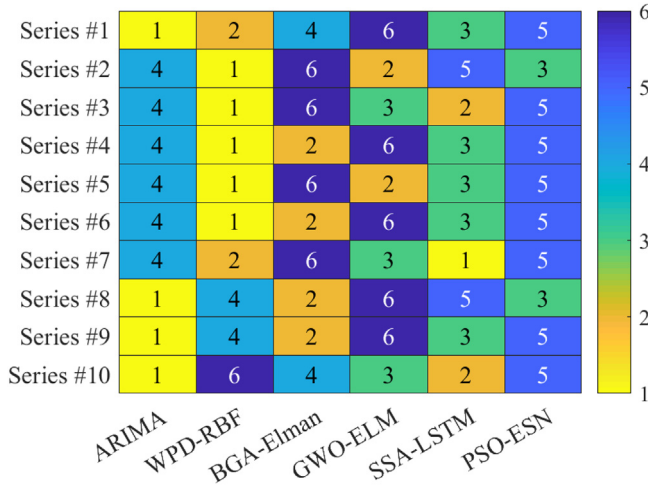


FIG. 2.30 – Forecasting performance ranks of six models.

2.8 Robot Anti-Collision Monitoring and Control Based on Navigation Position Forecasting

Robots are widely used in industry, hospitals, the military, and other places, and these robots have high requirements for robustness. For example, the fall prevention assist robot in hospital needs to provide accurate localization information timely (Namba and Yamada, 2018). The localization error may make mobile robots crash into patients, and lead to accidental injury. In a factory environment, the robot may collide with workers or obstacles if its location information is inaccurate.

When a mobile robot moves in a complex environment, the navigation device of the robot will inevitably be affected by strong light, metal, and other disturbances (Guo *et al.*, 2018). These influence factors are common and may lead to errors, gaps, and delays in the localization system. Thus, the wrong position information is provided to the mobile robot, which leads to the collision, deviation, or loss of the robot. It seriously affects the safe operation of the mobile robot in a complex environment (Barenji *et al.*, 2016).

In this chapter, the robot navigation position time series is used to build a prediction model, which can obtain the robot position information the next time. The predicted value is compared with the measured value. If the error of the two values exceeds the error set value, the predicted value is taken as the robot position information the next time. Otherwise, the measured value of the robot is taken as the position information of the robot at the next moment. Then, the position information of the robot at the next moment is stored in the position time series, and the position time series is updated to realize the calibration and prediction of the measured position of the robot navigation device until the robot stops moving.

The position forecasting model in this chapter can provide more information for anti-collision monitoring. The time-to-contact can be estimated by the prediction

position. The robot navigation system can achieve dynamic anti-collision control with time-to-contact.

2.9 Conclusions

This chapter illustrates several robot position forecasting models. Three different types of methods are introduced, including statistical methods, intelligent methods, and deep learning methods. Through decomposition, feature selection, and optimization, the prediction performance of the model is improved. The experimental conclusion can be drawn as follows:

- (a) The decomposition, feature selection, and optimization algorithms can improve the forecasting effect of models. The decomposition algorithms can decompose the original series into more predictable subseries. The feature selection algorithms can generate feature subsets. The subset contains enough information for prediction and prevents the curse of dimensionality. The optimization algorithms can provide globally optimized parameters for forecasting models.
- (b) The comprehensive model comparison study with 10 different position data indicates the ARIMA and WPD-RBF models are the best models. The deep learning models are not suitable for position-series forecasting. This phenomenon may be due to the overfitting of the deep learning models.

The applications for industrial and mobile robots based on the position prediction model are analyzed at last. The position prediction methods can enhance the robustness of the robot's navigation and anti-collision control.

References

- Alom M. Z., Taha T. M., Yakopcic C., Westberg S., Sidike P., Nasrin M. S., Asari V. K. (2019) A state-of-the-art survey on deep learning theory and architectures, *Electronics* **8**(3), 292.
- Barenji A. V., Barenji R. V., Hashemipour M. (2016) Flexible testing platform for employment of RFID-enabled multi-agent system on flexible assembly line, *Adv. Eng. Softw.* **91**, 1-11.
- Chen P., Pedersen T., Bak-Jensen B., Chen Z. (2009) ARIMA-based time series model of stochastic wind power generation, *IEEE Trans. Power Syst.* **25**(2), 667-676.
- Cho T.-H., Hwang H.-R., Lee J.-H., Lee I.-S. (2018) Comparison of intelligent methods of SOC estimation for battery of photovoltaic system, *Int. J. Adv. Comput. Sci. App.* **9**(9), 49-56.
- Contreras J., Espinola R., Nogales F. J., Conejo A. J. (2003) ARIMA models to predict next-day electricity prices, *IEEE Trans. Power Syst.* **18**(3), 1014-1020.
- DiGiampaolo E., Martinelli F. (2018) A robotic system for localization of passive UHF-RFID tagged objects on shelves, *IEEE Sens. J.* **18**(20), 8558-8568.
- Ding S., Xu X., Zhu H., Wang J., Jin F. (2011) Studies on optimization algorithms for some artificial neural networks based on genetic algorithm (GA), *J. Comput.* **6**(5), 939-946.
- Dorffner G. (2001) *Artificial Neural Networks-ICANN 2001: International Conference Vienna, Austria, August 21-25, 2001 Proceedings* (Vol. 2130). Springer Science & Business Media.
- Du K.-L., Swamy M. N. (2013) *Neural networks and statistical learning*. Springer Science & Business Media.
- Duan Z., Liu H. (2019) An evolution-dependent multi-objective ensemble model of vanishing moment with adversarial auto-encoder for short-term wind speed forecasting in Xinjiang wind farm, China, *Energy Convers. Manage.* **198**, 111914.

- Eliakim I., Cohen Z., Kosa G., Yovel Y. (2018) A fully autonomous terrestrial bat-like acoustic robot, *PLoS Comput. Biol.* **14**(9), e1006406.
- Elnagar A. (2011) Prediction of future configurations of a moving target in a time-varying environment using an autoregressive model, *Intell. Control Autom.* **2**(04), 284–292.
- Ganjdoost H., Mousavi S. J., Soroush A. (2016) Adaptive network-based fuzzy inference systems coupled with genetic algorithms for predicting soil permeability coefficient, *Neural Process. Lett.* **44**(1), 53–79.
- Garcia R. C., Contreras J., Van Akkeren M., Garcia J. B. C. (2005) A GARCH forecasting model to predict day-ahead electricity prices, *IEEE Trans. Power Syst.* **20**(2), 867–874.
- Gheyas I. A., Smith L. S. (2011) A novel neural network ensemble architecture for time series forecasting, *Neurocomputing* **74**(18), 3855–3864.
- Guo S., Fang T.-T., Song T., Xi F.-F., Wei B.-G. (2018) Tracking and localization for omni-directional mobile industrial robot using reflectors, *Adv. Manuf.* **6**(1), 118–125.
- Heming T., Jianwen W., Wenfu X., Wang Z., Yibo Y., Haipeng L., Weiren S. (2014, 29 June–4 July 2014). *Automated system of smart ammeter verification based on mobile robot*. In: Paper presented at the Proceeding of the 11th World Congress on Intelligent Control and Automation.
- Hol E. M. (2013) *Empirical studies on volatility in international stock markets* (Vol. 6). Springer Science & Business Media.
- Jia Y., Mo Y., Wang W., Guo S., Ma L. (2021) *Deep-Sea Source Ranging Method Using Modified General Regression Neural Network*. In: Paper presented at the 2021 OES China Ocean Acoustics (COA).
- Khosravi A., Nahavandi S., Creighton D. (2013) Quantifying uncertainties of neural network-based electricity price forecasts, *Appl. Energy* **112**, 120–129.
- Kostavelis I., Charalampous K., Gasteratos A., Tsotsos J. K. (2016) Robot navigation via spatial and temporal coherent semantic maps, *Eng. Appl. Artif. Intell.* **48**, 173–187.
- Lathi B. P., Green R. A. (2005) *Linear systems and signals* (Vol. 2). Oxford University Press, New York.
- Li C., Tang X. (2016) *Proceedings in Adaptation, Learning and Optimization*. Paper presented at the IES.
- Li K., Xue Y., Cui S., Niu Q., Yang Z., Luk P. (2017) *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration: International Conference on Life System Modeling and Simulation, LSMS 2017 and International Conference on Intelligent Computing for Sustainable Energy and Environment, ICSEE 2017, Nanjing, China, September 22–24, 2017, Proceedings, Part III* (Vol. 763). Springer.
- Lian J., Dong P., Zhang Y., Pan J., Liu K. (2020) A novel data-driven tropical cyclone track prediction model based on CNN and GRU with multi-dimensional feature selection, *IEEE Access* **8**, 97114–97128.
- Lin X., Yang Z., Song Y. (2009) Short-term stock price prediction based on echo state networks, *Expert Syst. Appl.* **36**(3), 7313–7317.
- Liu H. (2020) *Smart Cities: Big Data Prediction Methods and Applications*. Springer Nature.
- Liu H., Duan Z. (2020) Corrected multi-resolution ensemble model for wind power forecasting with real-time decomposition and Bivariate Kernel density estimation, *Energy Convers. Manag.* **203**, 112265.
- Liu H., Duan Z., Chen C. (2019) A hybrid framework for forecasting PM2. 5 concentrations using multi-step deterministic and probabilistic strategy, *Air Qual. Atmos. Health* **12**(7), 785–795.
- Liu H., Stoll N., Junginger S., Thurow K. (2015) *New localization strategy for mobile robot transportation in life science automation using StarGazer sensor, time series modeling and Kalman filter processing*. Paper presented at the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO).
- Liu H., Xu Y., Wu X., Lv X., Zhang D., Zhong G. (2019) *Big data forecasting model of indoor positions for mobile robot navigation based on apache spark platform*. Paper presented at the 2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA).
- Liu S. M., Brorsen B. W. (1995) Maximum likelihood estimation of a GARCH-stable model, *J. Appl. Econom.* **10**(3), 273–285.

- Luo B., Miao S. (2019) *A Literature Survey on Electricity Price Forecasting in Deregulated Markets*. Paper presented at the 2019 IEEE Sustainable Power and Energy Conference (iSPEC).
- Luo R. C., Hsiao T. J. (2019) Dynamic wireless indoor localization incorporating with an autonomous mobile robot based on an adaptive signal model fingerprinting approach, *IEEE Trans. Ind. Electron.* **66**(3), 1940–1951.
- Man Z., Lee K., Wang D., Cao Z., Miao C. (2011) *A modified ELM algorithm for single-hidden layer feedforward neural networks with linear nodes*. Paper presented at the 2011 6th IEEE conference on industrial electronics and applications.
- Moskalenko V., Moskalenko A., Korobov A., Semashko V. (2019) The model and training algorithm of compact drone autonomous visual navigation system. *Data* **4**(1), 4.
- Namba T., Yamada Y. (2018) Risks of deep reinforcement learning applied to fall prevention assist by autonomous mobile robots in the hospital, *Big Data Cogn. Comput.* **2**(2), 13.
- Onieva E., Santos I., Osaba E., Quintián H., Corchado E. (2015) *Hybrid Artificial Intelligent Systems: 10th International Conference, HAIS 2015, Bilbao, Spain, June 22–24, 2015, Proceedings* (Vol. 9121). Springer.
- Ren L., Liu Y., Rui Z., Li H., Feng R. (2009) *Application of Elman neural network and MATLAB to load forecasting*. Paper presented at the 2009 International Conference on Information Technology and Computer Science.
- Saeedvand S., Aghdasi H. S., Baltes J. (2018) Novel lightweight odometric learning method for humanoid robot localization, *Mechatronics* **55**, 38–53.
- Sanchez-Garcia A. J., Rios-Figueroa H. V., Garnier H., Quintana-Carapia G., Rechy-Ramirez E. J., Marin-Hernandez A. (2018) Predicting collisions: time-to-contact forecasting based on probabilistic segmentation and system identification, *Adv. Robot.* **32**(8), 426–442.
- Schalkoff R. J. (2007) Pattern recognition. *Wiley Encyclopedia of Computer Science and Engineering*.
- Sheinker A., Ginzburg B., Salomonski N., Frumkis L., Kaplan B.-Z., Moldwin, M. B. (2016) A method for indoor navigation based on magnetic beacons using smartphones and tablets, *Measurement* **81**, 197–209.
- Son D., Dong X., Sitti M. (2019) A simultaneous calibration method for magnetic robot localization and actuation systems, *IEEE Trans. Robot.* **35**(2), 343–352. <https://doi.org/10.1109/TRO.2018.2885218>.
- Tamura Y., Sato T., Ooe M., Ishiguro M. (1991) A procedure for tidal analysis with a Bayesian information criterion, *Geophys. J. Int.* **104**(3), 507–516.
- Teal R., Tubana B., Girma K., Freeman K., Arnall D., Walsh O., Raun W. (2006). In-season prediction of corn grain yield potential using normalized difference vegetation index, *Agron. J.* **98**(6), 1488–1494.
- Thong-un N. (2016) A linear least square approach for navigation using ultrasonic waves, *ECTI Trans. Electr. Eng. Electron. Commun.* **14**(1), 65–74.
- Torres-Solis J., Falk T. H., Chau T. (2010) A review of indoor localization technologies: towards navigational assistance for topographical disorientation. In: *Ambient Intelligence*. IntechOpen.
- Xydas E., Qadrdan M., Marmaras C., Cipcigan L., Jenkins N., Ameli H. (2017) Probabilistic wind power forecasting and its application in the scheduling of gas-fired generators, *Appl. Energy* **192**, 382–394.
- Yamada Y., Ito K., Tsuji T., Otani K., Kobayashi R., Watanabe Y., Hiryu S. (2019) Ultrasound navigation based on minimally designed vehicle inspired by the bio-sonar strategy of bats. *Adv. Robot.* **33**(3–4), 169–182.
- Yin S., Liu H., Duan Z. (2021) Hourly PM2.5 concentration multi-step forecasting method based on extreme learning machine, boosting algorithm and error correction model, *Digit. Signal Process.* 103221.
- Zong C., Nie J.-Y., Zhao D., Feng Y. (2015) Natural language processing and Chinese computing.

Chapter 3

Mobile Robot Power Time Series Predictive Control

3.1 Introduction

Power management and health monitoring are important for the robot control system, which can guarantee the safety and continuous working state of mobile robots. On the one hand, the endurance of mobile robots is mainly limited by electricity when working in high-load conditions for a long-term period. On the other hand, the health condition of the machine is also important. For example, stoppages may be caused by the power failure of the robot, if the robot is stuck in the running production line (Pearce *et al.*, 2018). Therefore, the power time series forecasting for the mobile robot is significant to ensure the normal operation and maintenance of manufacturing.

Aiming at realizing the mobile robot's recharging safely and efficiently, many studies have focused on the issue of recharging control and optimization. H. Liu *et al.* (2013) presented an improved artificial immune algorithm to select the best locations for recharging stations in a laboratory environment. The transportation distance and the object weight are considered in the optimization model. Arvin *et al.* (2018) proposed a probabilistic model for a robot swarm, which took the swarm size, robot task, and charging area into consideration. Many long-term experiments indicate that the probabilistic model has enough accuracy to support the perpetual operation of the mobile robot system. Zou *et al.* (2018) focused on battery management in robotic mobile fulfillment systems and built a semi-open queueing network to estimate the battery recharging and swapping strategies. These works mainly manage the power of mobile robots from the aspect of power recharging, which is also an important part of power management.

Besides, there are several approaches have been used for mobile robot power time series modeling. The prediction of battery power consumption is an important determinant of the demand for recharging robots. Hamza and Ayanian (2017) proposed a forecasting framework for onboard mobile robot power. Different tasks are mapped to the power consumption by nonlinear relationships. A neural network is trained to fit this nonlinear relationship. The experimental results show the

effectiveness of recurrent neural networks in power time series prediction. H. Liu *et al.* (2014) predicted and managed the battery voltage of the mobile robot with a new model. The wavelet decomposition method is selected to decompose the original voltage data into sub-layers. Each sub-layer is predicted by an adaptive network-based fuzzy inference system. Experiments prove the hybrid forecasting framework has a better prediction accuracy. Pentzer *et al.* (2014) estimated the energy consumption of mobile robots using terrain and kinematic parameters. The extended Kalman filter is used to predict the instantaneous rotation center position and the recursive least square is used to estimate power model parameters.

The power time series modeling methods need fewer parameters, compared to the methods using voltage, temperature, load current, and battery resistance parameters. Benefiting from the intelligent methods, the power time series modeling methods can achieve better forecasting accuracy. Power time series modeling can help the health monitoring of mobile robots. It can further improve the security of multiple mobile robot systems.

The research framework of power time series modeling mainly includes three parts: the statistical methods, the intelligent methods, and the deep learning methods, which are shown in figure 3.1.

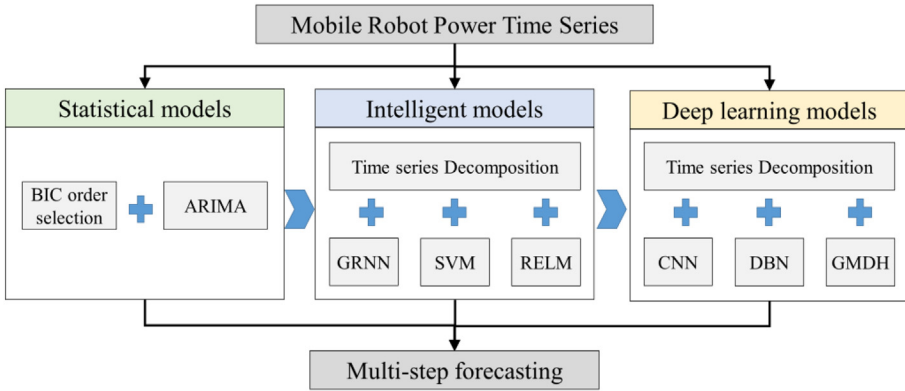


FIG. 3.1 – The research framework of chapter 3.

3.2 Mobile Robot Power Time Series Measurement

The real-time power voltages of mobile robots and the power percentage of the working laptops in the system should be measured for power time series modeling. The power time series modeling system should give the alarm when the power consumption meets the critical threshold.

As for the mobile robot health monitoring system, more parameters of the mobile robot should be recorded, including the temperature, load current, battery resistance, and so on. The hardware devices and sensors used are listed as follows: the voltage recorder, thermometric instrument, and weighing sensor of the mobile

robot. The mobile robot health monitoring system should give an alarm when those indices are higher than the normal value.

Another important piece of hardware equipment in the power management system is the power recharging station, including a stationary recharging station and a mobile recharging robot. Different parts of the power management system are shown in figure 3.2.

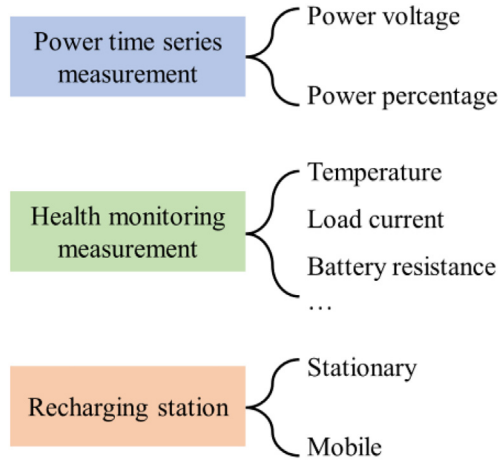


FIG. 3.2 – The measurement indices.

The power time series consists of the continuously measured power voltage or power percentage. Power time series modeling and forecasting can be used to manage power consumption and task arrangement for mobile robots. In the chapter, the power voltage time series is predicted with different methods. These methods are data-driven and do not require measuring device information.

3.3 Mobile Robot Power Time Series Uncertainty Analysis

The uncertainty of the power voltage time series mainly contains two aspects, including environmental uncertainty and measurement uncertainty.

(1) *The environmental uncertainty*

Environmental uncertainty can affect the power consumption of mobile robots. Different environmental characteristics include temperature, humidity, and environmental complexity.

Numerous experiments show that mobile robots have higher battery power consumption at high temperatures. The low-temperature environment may cause the power of the mobile robot to shut down before the battery runs out.

Environmental humidity can influence the working state of key electronic equipment on mobile robots. Power consumption and recharging are affected by abnormal electronic equipment. The environmental complexity can affect the robot's navigation and path-planning system. A complex surrounding environment improves the computational load of robot navigation and path planning system, and further increases the power voltage consumption of mobile robots. Notably, a sufficiently complex surrounding environment can make the robot system halted, because the robot navigation system cannot work out it and the power consumption of the calculating system can be relatively high.

(2) *Measurement uncertainty*

The measurement uncertainty can influence the power voltage time series measurement of mobile robots. The measurement uncertainty can generate outliers in the power voltage time series and affect the forecasting results.

The equipment failure of measuring sensors can generate meaningless outliers in the power voltage time series. Those outliers provide wrong information about the power consumption of mobile robots. The power time series forecasting models are affected by the outliers and get incorrect forecasting results.

The measuring accuracy of hardware sensors also belongs to the measurement uncertainty of mobile robots. The measuring error causes the power voltage time series to move upward or downward, which makes the forecasting results of power consumption late or early.

3.4 Mobile Robot Power Time Series Statistical Forecasting Method

The statistical method is selected to forecast the power time series in the section. The Autoregressive Integrated Moving Average (ARIMA) model, a typical statistical method, is selected to fit and forecast the power time series. M. H. Amini *et al.* (2016) proposed an ARIMA-based method to forecast the recharging demand for electric vehicles. M. Amini *et al.* (2015) proposed an ARIMA model to predict EV recharging demand, where the parking lots were considered. In the section, the ARIMA method is adopted to forecast the power time series for mobile robots.

3.4.1 *Experimental Design*

(1) *Data description*

The power voltage time series is measured in past experiments of mobile robots. The sampling interval of power voltage data is 1 s. 1000 power voltage data are used in the experiment, where 1st–900th data are utilized to train the ARIMA model, and 901st–1000th data are used as the testing set. As shown in figure 3.3, the original power voltage time series is a non-stationary series. The ARIMA method demands a one-time difference for the original power voltage data, and the parameter D of the ARIMA model is 1.

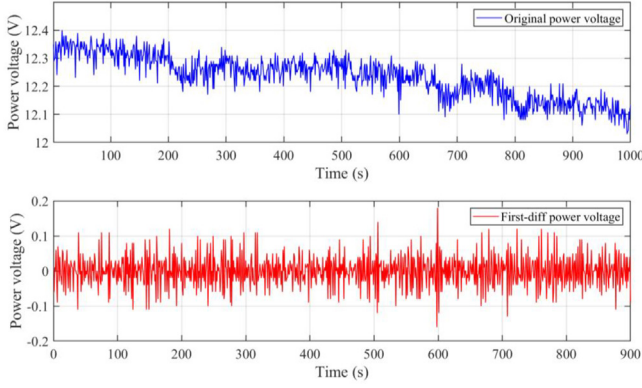


FIG. 3.3 – Power voltage time series.

(2) Evaluation indices

In this section, the forecasting performance can be evaluated by Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE). The equations of these indices are presented as formula (2.9) in section 2.4.1.

3.4.2 Modeling Steps

(1) Data preprocessing

Before establishing the ARIMA (p, d, q) model, the data series should be stationary. The augmented Dickey-Fuller (ADF) test is widely used for stationarity tests. According to the ADF test, the original power voltage time series is non-stationary. So the original power voltage data should be differenced one-time. The results are shown in figure 3.3. So the parameter d of the ARIMA model is 1.

(2) Parameter estimation

After the ADF test, the ARIMA model is established preliminarily according to the autocorrelation function and partial correlation function. The results of these two functions are shown in figure 3.4.

According to figure 3.4, the autocorrelation function has an obscure censored feature, and the partial correlation function has a trailing feature. So the ARIMA $(p, 1, q)$ model needs further confirmation. The auto-regressive parameter p and moving averaging parameter q should be determined by the Bayesian Information Criterion (BIC).

The BIC criterion can evaluate the performance of statistical models comprehensively. The BIC criterion can be defined as follows (Nichiforov *et al.*, 2017):

$$\text{BIC}(n) = \ln(n)k - 2 \ln(\hat{L}) \quad (3.1)$$

where $\hat{L}$ is the likelihood, n is the number of observations, k is the number of the parameters.

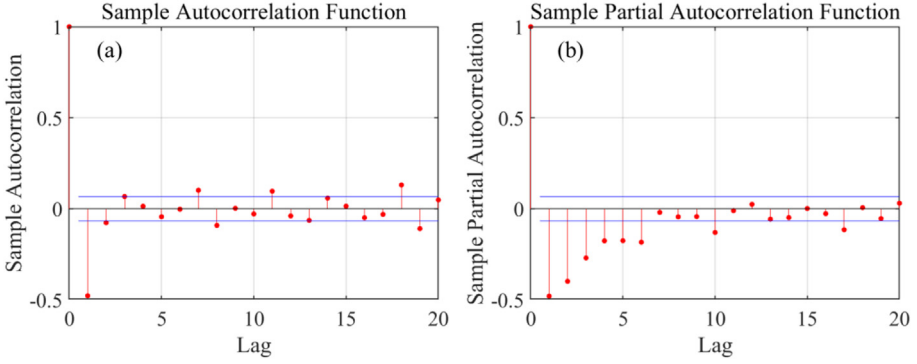


FIG. 3.4 – Autocorrelation function and partial correlation function of the differenced power voltage series: (a) autocorrelation function, (b) partial correlation function.

The auto-regressive parameter p and moving averaging parameter q both range within $[0,10]$. The ARIMA $(p,1,q)$ models fit the original power voltage time series. The BIC criterion results of different ARIMA models are shown in figure 3.5. It can be seen from figure 3.5 that the best ARIMA model is ARIMA $(2,1,1)$.

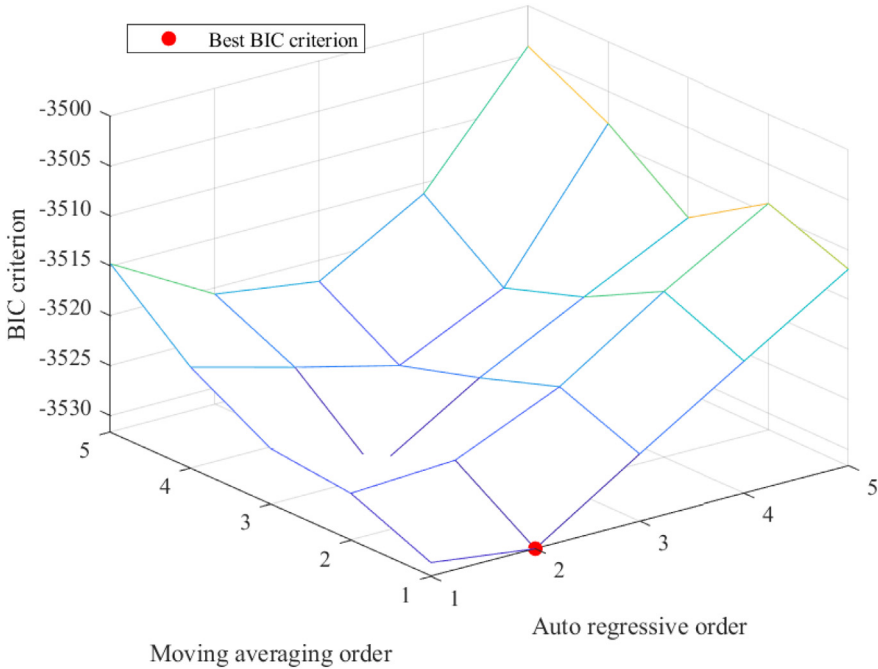


FIG. 3.5 – BIC criterion results of different ARIMA models.

(3) *Parameter determination*

The primary expression of ARIMA (p,d,q) is given as follows (Nichiforov *et al.*, 2017):

$$\begin{cases} \Phi(B)\nabla^d X_t = \Theta(B)\varepsilon_t \\ \Phi(B) = 1 - f_1 B - L - f_p B^p \\ \Theta(B) = 1 - \theta_1 B - L - \theta_q B^q \end{cases} \quad (3.2)$$

where $\nabla X_t = (1 - B)^d$ represents the ARIMA (p,d,q) as a d -differenced model. $\Phi(B)$ is the autoregressive polynomial, $\Theta(B)$ is the moving average polynomial of the ARIMA (p,d,q) model. ε_t is random error.

After selecting the structure of the ARIMA model, the parameters should be determined by the maximum likelihood method. In the study, the autoregressive coefficient is -0.0953 and -0.1201 . The moving average coefficient is -0.8159 and the random error is 0.0011 . So, the ARIMA $(2,1,1)$ model is determined as follows:

$$(1 + 0.0953B + 0.1201B^2)(1 - B)X_t = (1 - 0.8159B)\varepsilon_t \quad (3.3)$$

The above equation can be solved as follows:

$$X_t = -1.0953X_{t-1} - 0.0248X_{t-2} + 0.1201X_{t-3} + \varepsilon_t - 0.8159\varepsilon_{t-1} \quad (3.4)$$

where X_t, X_{t-1}, X_{t-2} is the observation in time $t, t-1, t-2$, $\varepsilon_t, \varepsilon_{t-1}$ is the innovation in time $t, t-1$. The innovation ε_t obeys Gaussian distribution.

(4) *Forecast*

In the ARIMA model, the final forecasting results are obtained by minimizing mean square error, which is given as follows (de Oliveira and Oliveira, 2018):

$$\min \left\{ E[\hat{e}_t(l)]^2 \right\} = \min \left\{ E[X_{t+l} - \hat{X}_t(l)]^2 \right\} \quad (3.5)$$

where X_{t+l} is the observation value for time $t+l$, $\hat{X}_t(l)$ is the forecasting value for time $t+l$. The minimum mean square error forecasting is presented as follows (de Oliveira and Oliveira, 2018):

$$\hat{X}_t(l) = E[X_{t+l}|X_t, X_{t-1}, \dots, X_1] \quad (3.6)$$

The minimum mean square error 1-step forecasting equation of the ARIMA model can be obtained as follows:

$$\hat{X}_t(1) = E[X_{t+1}] = -1.0953X_{t-1} - 0.0248X_{t-2} + 0.1201X_{t-3} - 0.8159\varepsilon_{t-1} \quad (3.7)$$

3.4.3 Forecasting Results¹

The forecasting results of the ARIMA statistical model are shown in figures 3.6–3.8.

The evaluation indices of the ARIMA forecasting results are shown in table 3.1.

¹Part of text in this section is re-used from section 8.3.4 of Springer Nature book “*Smart Cities: Big Data Prediction Methods and Applications*” (Liu, 2020) under permission.

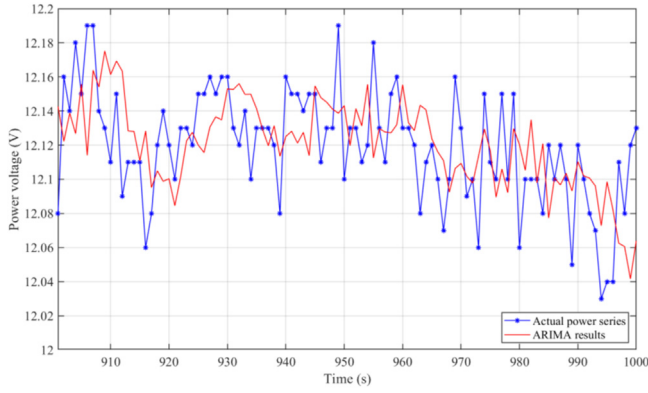


FIG. 3.6 – The forecasting results of power voltage time series in 1-step.

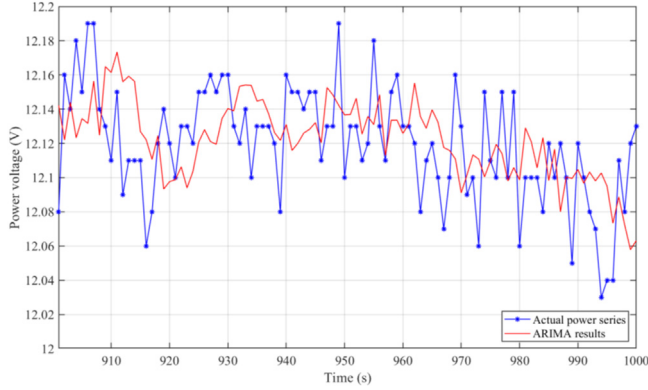


FIG. 3.7 – The forecasting results of power voltage time series in 3-step.

The comparative analysis of ARIMA forecasting results is shown in section 3.7.1.

3.5 Mobile Robot Power Time Series Intelligent Forecasting Method

Intelligent methods can predict more accurately when compared to traditional statistical methods. In this section, three widely used intelligent methods are selected to forecast mobile robot power time series, including Generalized Regression Neural Network (GRNN), Support Vector Machine (SVM), and Regularized Extreme Learning Machine (RELM).

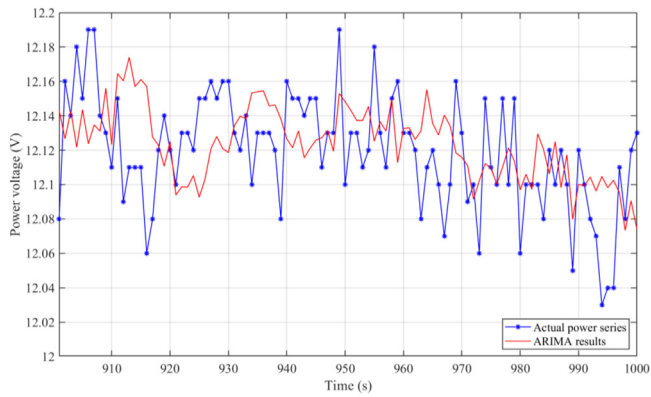


FIG. 3.8 – The forecasting results of power voltage time series in 5-step.

TAB. 3.1 – The evaluation indices of ARIMA model.

Model	Step	MAE (V)	MAPE (%)	RMSE (V)
ARIMA (2,1,1)	1	0.0329	0.2718	0.0387
	3	0.0318	0.2624	0.0368
	5	0.0312	0.2575	0.0357

3.5.1 Experimental Design

(1) Data description

The power voltage time series is measured in past experiments of mobile robots. The sampling interval of power voltage data is 1 s. 1000 power voltage data are used in the experiment, where 1st–900th data are utilized to train the hybrid intelligent models, and 901st–1000th data are used as the testing set. The power voltage time series is shown in figure 3.9.

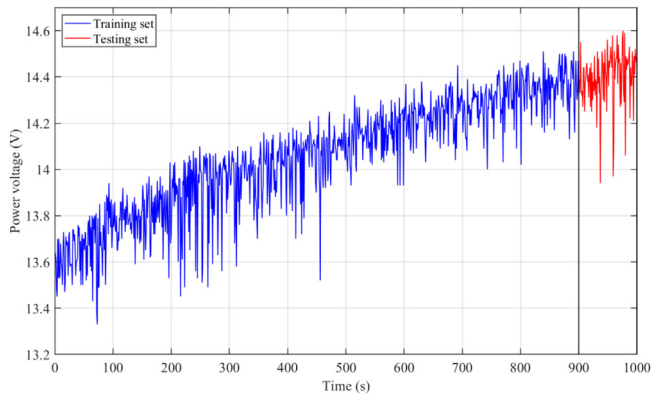


FIG. 3.9 – Original power voltage time series for intelligent forecasting experiments.

(2) Wavelet packet decomposition

Wavelet Packet Decomposition (WPD) is a variant of wavelet decomposition. The binary tree of the WPD with three decomposition layers is shown in figure 3.10. The WPD can obtain a comprehensive information division of the original time series by decomposing the series into approximation component (A) and detailed component (D) of each layer.

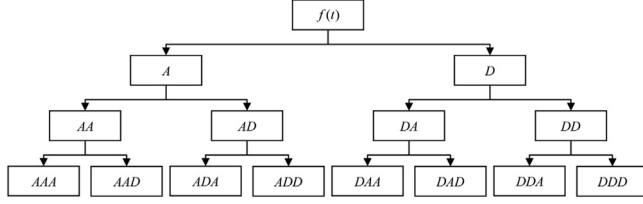


FIG. 3.10 – The binary trees of WPD.

The WPD consists of several filter banks. The calculation is defined as follows (Gokhale and Khanduja, 2010):

$$y[n] = (x * F)[n] = \sum_{k=-\infty}^{\infty} x[k]F[n - k] \quad (3.8)$$

where F is the impulse response of the corresponding filter, and x is the original power voltage time series.

(3) Generalized regression neural network

Specht (1991) proposed the GRNN in 1991. The GRNN is a modified network of Radial Basis Function (RBF) neural network. The GRNN structure contains four layers: input layer, pattern layer, summation layer, and output layer. The structure of GRNN is shown in figure 3.11.

The mathematical representation of GRNN can be given as follows (Specht, 1991):

$$Y(x) = \frac{\sum_{K=1}^N y_k K(x, x_k)}{\sum_{K=1}^N K(x, x_k)} \quad (3.9)$$

where $Y(x)$ is the prediction value of input x . y_k is the activation weight for the pattern layer neuron at k . $K(x, x_k)$ is the RBF kernel of GRNN, which is formulated as follows (Specht, 1991):

$$K(x, x_k) = e^{-d_k/2\delta^2}, \quad d_k = (x - x_k)^T (x - x_k) \quad (3.10)$$

where d_k is the squared Euclidean distance between the training sample x_k and input x .

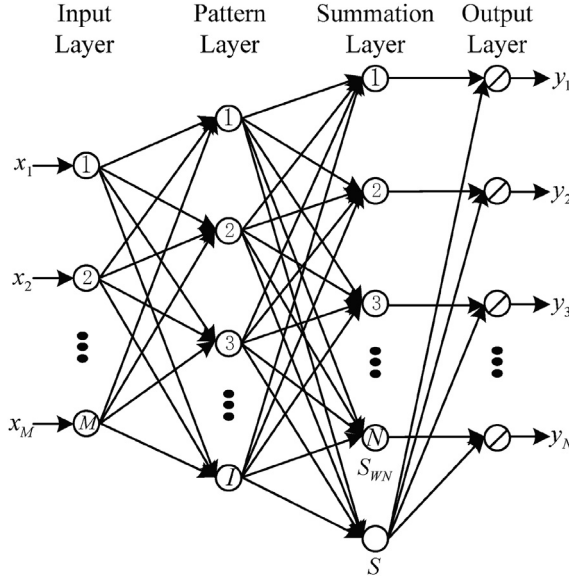


FIG. 3.11 – Structure of GRNN.

(4) *Support vector machine*

The SVM comes from the dichotomy problem (H. Liu *et al.*, 2019). Smola and Schölkopf (2004) applied the SVM to the regression problem. The Support Vector Regression (SVR) can be used in the power time series forecasting problem.

The solving process of SVR can be expressed as the following optimization problem (Ghelardoni *et al.*, 2013):

$$\begin{aligned} \min & \left\{ \frac{\|\omega\|^2}{2} + C \sum_{i=1}^M (\xi_i^1 + \xi_i^2) \right\} \\ \text{s.t.} & \begin{cases} y_i - (\omega \cdot x_i + b) \leq \varepsilon + \xi_i^1 \\ (\omega \cdot x_i + b) - y_i \leq \varepsilon + \xi_i^2 \\ \xi_i^1 \geq 0, \xi_i^2 \geq 0, i = 1, 2, \dots, M \end{cases} \end{aligned} \quad (3.11)$$

where ξ_i^1 and ξ_i^2 are slack variables. The penalty parameter C is used to coordinate the empirical risk and structural risk of the model.

(5) *Regularized extreme learning machine*

Traditional Extreme Learning Machine (ELM) only considers empirical risk minimization, which is prone to the over-fitting problem (Sun and Liu, 2016). The Regularized Extreme Learning Machine (RELM) considers both empirical risk minimization and structural risk minimization (J. Wang *et al.*, 2018). This process in the RELM can be shown as follows (W. Deng *et al.*, 2009):

$$\min : \|\beta\|^2 + \sum_{i=1}^N \|\beta * h(x_i) - t_i\| \quad (3.12)$$

The equation that combines empirical risk minimization and structural risk minimization can be expressed below (W. Deng *et al.*, 2009):

$$\begin{aligned} \min : L &= \frac{1}{2} \left(\|\beta\|^2 + C \sum_{i=1}^N \|\xi_i\|^2 \right) \\ \text{s.t.} : h(x_i)\beta &= t_i^T - \xi_i^T, \quad i = 1, 2, \dots, N \end{aligned} \quad (3.13)$$

where ξ_i represents the output error. The Lagrangian function can solve the above problem. The output matrix can be obtained as follows:

$$\beta = H^T \left(HH^T + \frac{1}{C} \right)^{-1} T \quad (3.14)$$

Besides, there is little difference between RELM and ELM methods.

3.5.2 Modeling Steps

(1) Structure of the intelligent models

Three intelligent hybrid models are built to forecast power voltage time series in this section, including WPD-GRNN, WPD-SVM, and WPD-RELM. Those models have similar modeling steps, which are shown in figure 3.12.

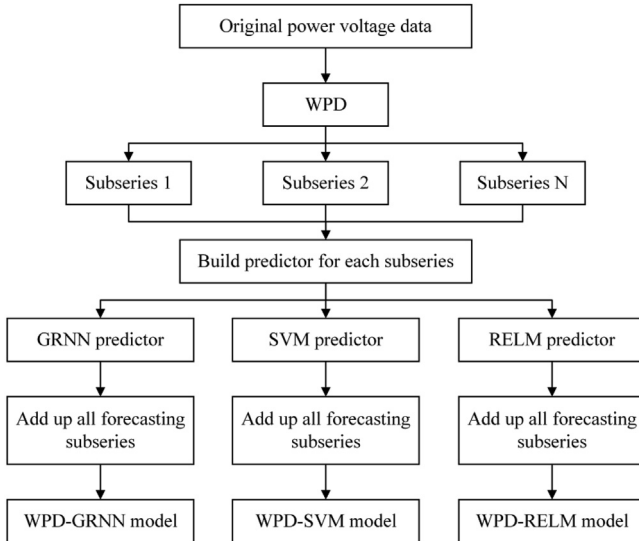


FIG. 3.12 – The modeling steps for intelligent models.

The measured power voltage time series is decomposed by the WPD method into eight subseries. The complexity of the original power voltage data is reduced in this way. The GRNN, SVM, and RELM are selected as the predictors to obtain forecasting results for each subseries. Different forecasting results of corresponding subseries are summed up to get the final forecasting results of different models, including the WPD-GRNN, WPD-SVM, and WPD-RELM.

(2) *Data decomposition*

The original power voltage time series is too complex for intelligent methods to forecast. The original power voltage data series is decomposed into subseries by the WPD. The complexity of each subseries is much less than the original data. The results of data decomposition are shown in figure 3.13.

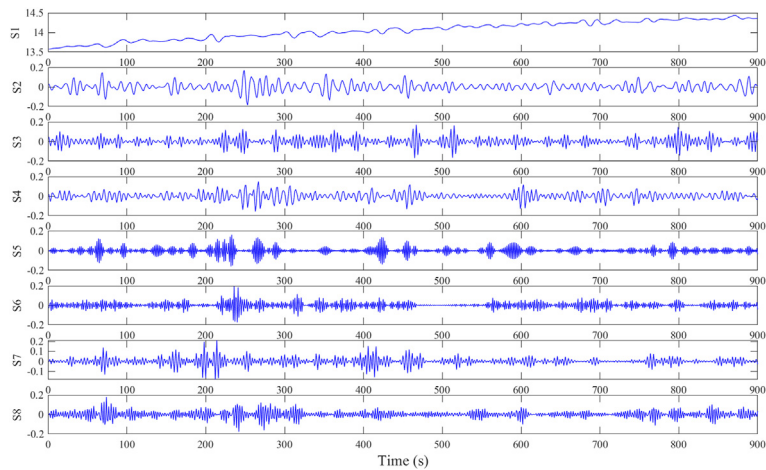


FIG. 3.13 – Subseries decomposed by the WPD.

As shown in figure 3.14, the power voltage data are more concentrated on the low-frequency band. The subseries S1 shows the main trend of the original power voltage time series.

(3) *Train and forecast*

After data decomposition, different subseries are used to train different predictors, including GRNN, SVM, and RELM. The parameter settings of different predictors are shown in table 3.2.

TAB. 3.2 – Parameter settings of different predictors.

GRNN	Smoothing factor δ	0.0775
	Penalty parameter C	0.5
SVM	Parameter gamma g	1
	Hidden layer size	20
RELM	Regularized term coefficient	0.0001

After different intelligent predictors are trained, they are used to forecast all the subseries. The forecasting results of different subseries are gathered to get the final results of different hybrid intelligent models, including WPD-GRNN, WPD-SVM, and WPD-RELM.

3.5.3 Forecasting Results

The 1-step, 3-step, and 5-step ahead forecasting results of the three intelligent models are shown in figures 3.14–3.16.

The evaluation indices of the three intelligent models are shown in table 3.3.

The forecasting results of different intelligent models are compared and analyzed in section 3.7.2.

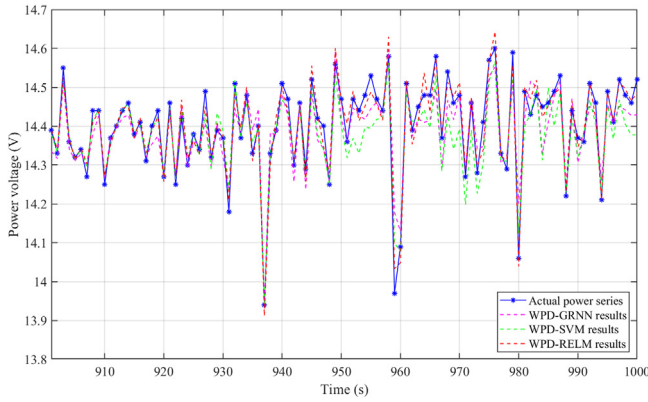


FIG. 3.14 – Forecasting results of intelligent models in 1-step.

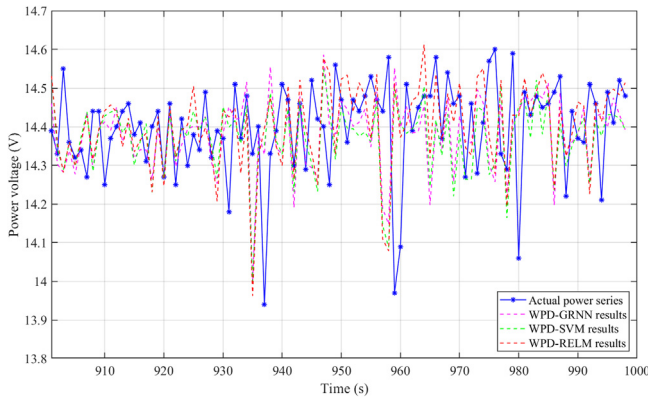


FIG. 3.15 – Forecasting results of intelligent models in 3-step.

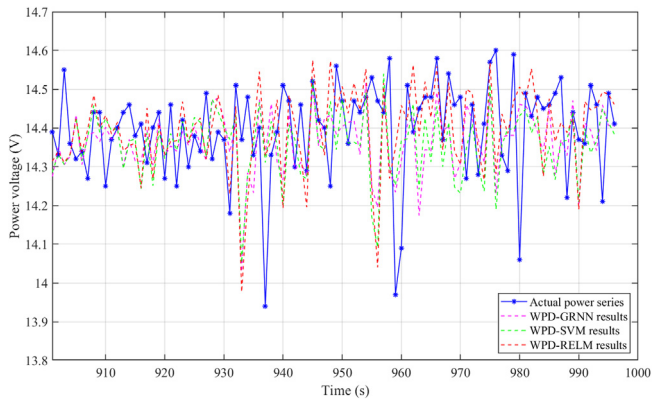


FIG. 3.16 – Forecasting results of intelligent models in 5-step.

TAB. 3.3 – The evaluation indices of intelligent models.

Model	Step	MAE (V)	MAPE (%)	RMSE (V)
WPD-GRNN	1	0.0549	0.3812	0.0504
	3	0.0901	0.6189	0.0734
	5	0.0454	0.3173	0.0947
WPD-SVM	1	0.0126	0.0875	0.0527
	3	0.0188	0.1289	0.0664
	5	0.0334	0.2335	0.0883
WPD-RELM	1	0.0008	0.0054	0.0266
	3	0.0056	0.0393	0.0468
	5	0.0187	0.1287	0.0690

3.6 Mobile Robot Power Time Series Deep Learning Forecasting Method

Deep learning has been prevailing in the recent few years. Compared with shallow methods, forecasting models based on deep learning can extract and learn more in-depth information about time series. In this section, three deep learning forecasting methods are employed to forecast mobile robot power time series, including Convolutional Neural Network (CNN), Deep Belief Network (DBN), and Group Method of Data Handling (GMDH).

3.6.1 Experimental Design

(1) Data description

The original power series used in the section is shown in figure 3.17, containing 1000 data in total. It was measured from a robot performing a specific task and thus shows a downward trend. The former 900 data are used to train the deep predictors, and the last 100 data are used to test the forecasting performance.

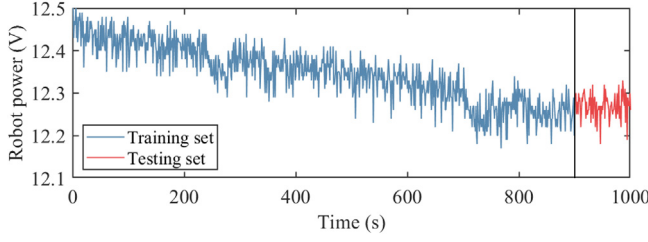


FIG. 3.17 – Original power voltage time series for deep learning forecasting experiments.

(2) Empirical wavelet transform

Different from the WPD, Empirical Wavelet Transform (EWT) can determine the number of decomposed subseries adaptively, leading to relatively few parameters. The decomposition layers and wavelets in WPD are not required to be set in advance. The EWT includes the empirical wavelet function and empirical scale function. They are defined as the band-pass filter and low-pass filter on each frequency band, which can be expressed as follows (Gilles, 2013):

$$\phi_n(\omega) = \begin{cases} 1, & |\omega| \leq (1 - \gamma)\omega_n \\ \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_n}(|\omega| - (1 - \gamma)\omega_n)\right)\right], & (1 - \gamma)\omega_n \leq |\omega| \leq (1 + \gamma)\omega_n \\ 0, & \text{otherwise} \end{cases} \quad (3.15)$$

$$\psi_n(\omega) = \begin{cases} 1, & (1 + \gamma)\omega_n \leq |\omega| \leq (1 - \gamma)\omega_{n+1} \\ \cos\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_{n+1}}(|\omega| - (1 - \gamma)\omega_{n+1})\right)\right], & (1 - \gamma)\omega_{n+1} \leq |\omega| \leq (1 + \gamma)\omega_{n+1} \\ \sin\left[\frac{\pi}{2}\beta\left(\frac{1}{2\gamma\omega_n}(|\omega| - (1 - \gamma)\omega_n)\right)\right], & (1 - \gamma)\omega_n \leq |\omega| \leq (1 + \gamma)\omega_n \\ 0, & \text{otherwise} \end{cases} \quad (3.16)$$

(3) Convolutional neural network

In recent years, CNN has been proved superior in many fields, especially image classification (Krizhevsky *et al.*, 2012). However, the CNNs are usually employed to process two-dimensional data. The one-dimensional power time series needs to be rationally converted. In the proposed model, the original input of CNN is the 25 past observations $X = [x_{t-25}, x_{t-24}, \dots, x_{t-2}, x_{t-1}]$. They are converted to a 5×5 matrix so that the CNN can be constructed to learn the characteristics of the power series. The converted two-dimensional matrix can be shown as follows:

$$X = \begin{bmatrix} x_{t-25} & \cdots & x_{t-21} \\ x_{t-20} & \cdots & x_{t-16} \\ x_{t-15} & \cdots & x_{t-11} \\ x_{t-10} & \cdots & x_{t-6} \\ x_{t-5} & \cdots & x_{t-1} \end{bmatrix} \quad (3.17)$$

(4) Deep belief network

An important issue in deep learning is developing appropriate training methods. With the increase in the number of layers in the traditional neural network, it is difficult to find the optimal solution by using stochastic gradient descent. In the process of backpropagation, gradient dispersion or gradient saturation may lead to unsatisfactory results in the model.

To solve the above problem, Geoffrey Hinton proposed DBN in 2006 (Hinton *et al.*, 2006). It is established on an energy-based model, called Restricted Boltzmann Machine (RBM). The RBM has a visible layer and a hidden layer. The layers are fully connected, and there is no connection within each layer. The value of the node is 0 or 1, which is a binary graph. It is a probabilistic generative model, defined as follows (Hinton *et al.*, 2006):

$$p(v_i = 1|\mathbf{h}) = \sigma \left(a_i + \sum_j w_{i,j} h_j \right) \quad (3.18)$$

$$p(h_i = 1|\mathbf{v}) = \sigma \left(b_i + \sum_j w_{i,j} v_j \right) \quad (3.19)$$

where a_i and b_i are the biases of the visible layer and hidden layer.

The DBN is developed by stacking multiple RBMs. It includes two implementation steps: unsupervised pre-training and supervised fine-tuning. The greedy and layer-wise pre-training is achieved by contrastive divergence learning. After each layer is pre-trained, the model can be fine-tuned by backpropagation to converge to the optimum. In this way, a deep network can be effectively trained. The proposed DBN contains two hidden layers, which include 100 and 50 nodes, respectively.

(5) Group Method of Data Handling

The GMDH is a self-organized algorithm, that continuously increases the complexity of the model and selects the best solution according to external criteria. The most popular support function of GMDH is the Kolmogorov-Gabor polynomial, which can be expressed as below (Madala and Ivakhnenko, 1994):

$$\hat{y} = a_0 + \sum_{i=1}^m a_i x_i + \sum_{i=1}^m \sum_{j=1}^m a_{ij} x_i x_j + \sum_{i=1}^m \sum_{j=1}^m \sum_{k=1}^m a_{ijk} x_i x_j x_k + \cdots \quad (3.20)$$

where $a_0, a_i, a_{ij}, a_{ijk}$ are the coefficients estimated by the least square method.

After the first layer is formed, several neurons are selected as input for the next layer. This procedure successively performs to develop the model with optimal complexity.

3.6.2 Modeling Steps

(1) Structure of the deep learning models

Three models are built in this section, including the EWT-CNN, EWT-DBN, and EWT-GMDH. They have similar modeling flows, as shown in figure 3.18.

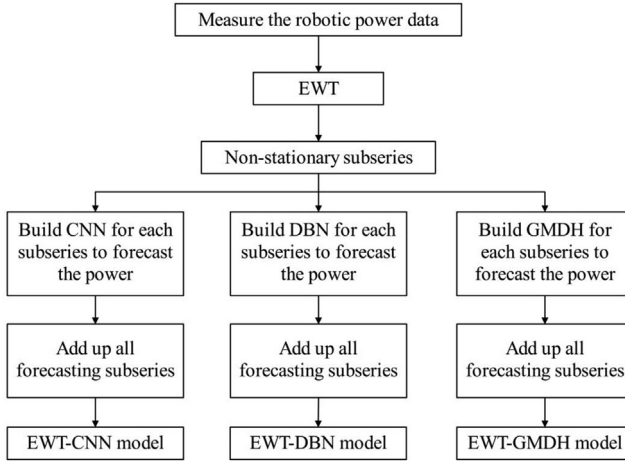


FIG. 3.18 – Modeling flows of power deep learning forecasting methods.

The measured power time series is decomposed into 8 non-stationary subseries by the EWT so that the forecasting ability is reduced. The number of subseries is determined adaptively by the Fourier segmentation. For each subseries, a CNN/DBN/GMDH model is constructed for forecasting. Then they are summed up to get the final forecasting result.

(2) Data decomposition

The original power series is decomposed by the EWT. Figure 3.19 shows the adaptive segmentation of the subseries. According to the segmentation of EWT, the original power series is divided into 14 parts in the Fourier domain.

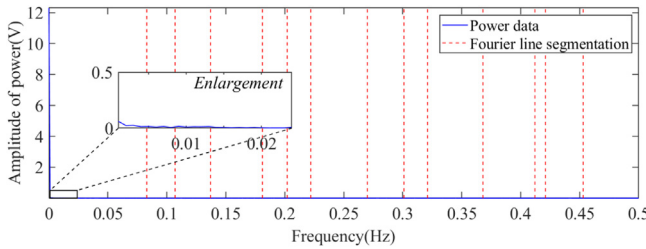


FIG. 3.19 – Segmentation of power series in Fourier domain.

Figure 3.20 presents the decomposed series. Apparently, the power data are more concentrated on the low-frequency band, shown as S1, which represents the main trend of the original series. For other subseries, the amplitudes are relatively small.

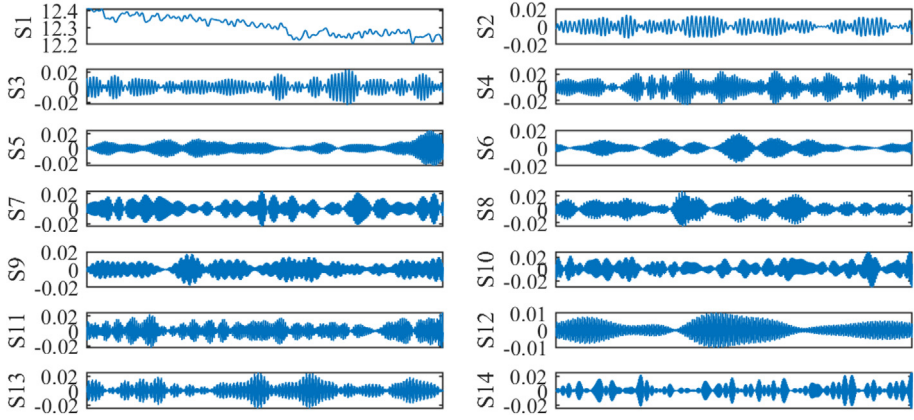


FIG. 3.20 – Subseries decomposed by the EWT.

(3) Train and forecast

For each subseries, a deep learning predictor (CNN/DBN/GMDH) is trained to forecast the future trend of the power time series. Some of their implementation details are presented in this section. The multi-step ahead forecasting of CNN and DBN can be achieved by the multi-input multi-output structure. However, the GMDH utilizes the recursive structure. On the basis of one-step-ahead prediction, the prediction result is used as the model input to realize the multi-step-ahead prediction.

Figure 3.21 plots the loss curves of CNNs during the training process. The maximum iterations are set as 1000. Apparently, the loss curves rapidly drop and slightly fluctuate in the end. This demonstrates that the CNNs have great convergence and are well-trained.

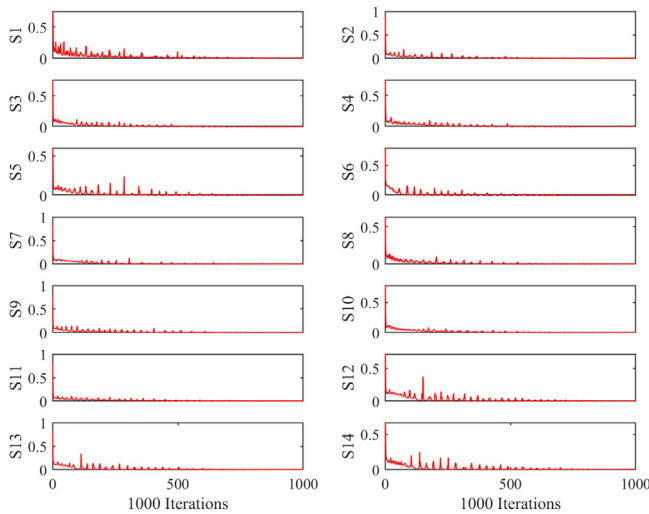


FIG. 3.21 – Loss curves of CNNs.

Besides, to ensure the computational efficiency of the GMDH, the number of layers and the maximum number of neurons in each layer need to be limited, which are set as 4 and 25, respectively. During the training process of the GMDH, the number of neurons in each layer is adaptively determined, as shown in table 3.4. Since the output of the GMDH is a single predicted value, the number of neurons in Layer 4 is always 1.

TAB. 3.4 – Number of neurons in the GMDH.

Subseries	Layer 1	Layer 2	Layer 3	Layer 4
#1	25	4	3	1
#2	25	25	18	1
#3	25	25	25	1
#4	25	5	4	1
#5	25	20	25	1
#6	25	21	25	1
#7	25	25	25	1
#8	22	15	25	1
#9	25	21	25	1
#10	18	6	4	1
#11	6	6	5	1
#12	25	25	25	1
#13	16	25	25	1
#14	25	25	25	1

3.6.3 Forecasting Results

The 1-step, 3-step, and 5-step ahead forecasting results of the three deep learning models are shown in figures 3.22–3.24.

The evaluation indices of corresponding forecasting results are shown in table 3.5.

The results of deep learning models will be compared and analyzed in section 3.7.3.

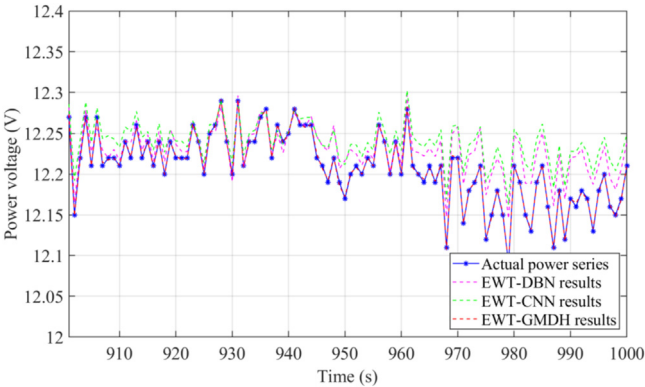


FIG. 3.22 – Forecasting results of deep learning models in 1-step.

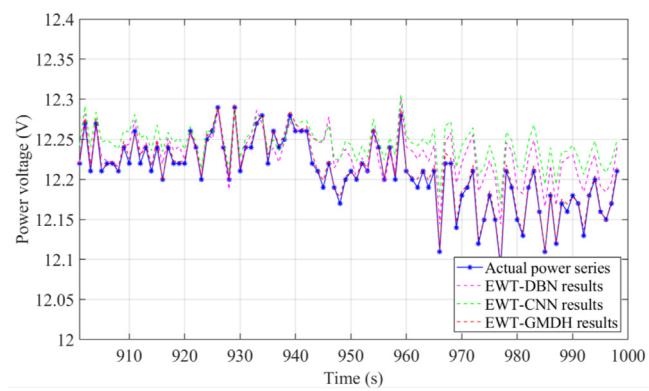


FIG. 3.23 – Forecasting results of deep learning models in 3-step.

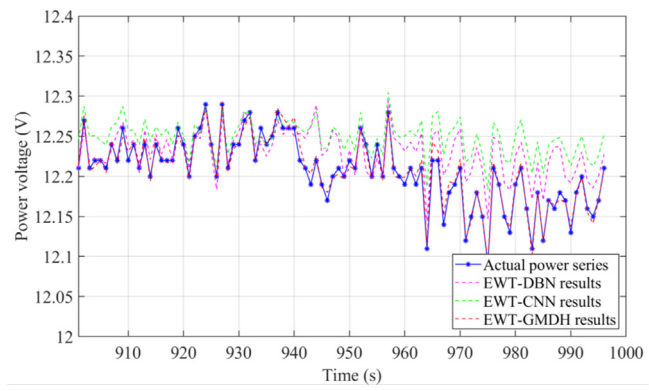


FIG. 3.24 – Forecasting results of deep learning models in 5-step.

TAB. 3.5 – The evaluation indices of deep learning models.

Model	Step	MAE (V)	MAPE (%)	RMSE (V)
EWT-DBN	1	0.0226	0.1856	0.0288
	3	0.0243	0.1991	0.0307
	5	0.0278	0.228	0.0348
EWT-CNN	1	0.0319	0.2617	0.0381
	3	0.0368	0.302	0.0436
	5	0.0418	0.3435	0.0496
EWT-GMDH	1	0.0015	0.0124	0.0018
	3	0.0041	0.0337	0.005
	5	0.0054	0.0442	0.0068

3.7 Comparative Analysis of Forecasting Performance

The forecasting results and evaluation indices of different models are analyzed and compared in the section.

3.7.1 Analysis of Statistical Methods

According to the results of statistical methods in section 3.4.3, the following conclusions can be drawn:

- (a) From figures 3.6–3.8, it can be seen that the ARIMA statistical model can only fit the basic trend of the original power voltage time series, but cannot forecast accurately. A single ARIMA model cannot meet the requirement of power time series forecasting for mobile robots.
- (b) According to table 3.1, the evaluation indices show that the ARIMA model has a good performance. This phenomenon can be explained as follows: the original power voltage time series varies very little from 14.0 V to 14.5 V. The power voltage changing range is only 0.5 V, which makes the value of all evaluation indices low.
- (c) From figures 3.6–3.8 and tables in section 3.4.3, it can be seen that the performance of the ARIMA statistical model is affected slightly by the forecasting steps. From 1-step to 5-step, the results of the ARIMA model are similar. This phenomenon proves that the ARIMA model has sufficient stability.

3.7.2 Analysis of Intelligent Methods

According to the results of intelligent models in section 3.5.3, the following conclusions can be drawn:

- (a) As shown in figures 3.14–3.16, all three intelligent models can forecast the power voltage series accurately, including the WPD-GRNN, WPD-SVM, and WPD-RELM models. The evaluation indices in table 3.3 also prove the effectiveness of those intelligent models.
- (b) The forecasting steps are the most influential factor in all three intelligent models. As shown in table 3.3, all models tend to become worse with the development of forecasting steps. It can be seen from figure 3.16 that the intelligent models cannot meet the accuracy requirement of power voltage time series in 5-step forecasting. The stability of the intelligent models needs to be improved.
- (c) It can be seen from table 3.3 that the WPD-RELM model has the best performance in 1-step, 3-step, and 5-step forecasting. Especially, the WPD-RELM model can forecast the power time series accurately in 1-step forecasting, the MAR, MAPE, and RMSE of it are 0.0008 V, 0.0054%, and 0.0266 V, respectively. Among three intelligent models, the WPD-RELM model outperforms the other two models. The improving percentages of WPD-RELM compared to other models are shown in table 3.6. Conclusively, the WPD-RELM has proved to be an effective and outstanding power time series forecasting model.

TAB. 3.6 – Improving percentages of the WPD-RELM model.

Improving percentages	P_{MAE} (%)	P_{MAPE} (%)	P_{RMSE} (%)
WPD-RELM <i>vs.</i> WPD-GRNN			
Step 1	98.543	98.583	47.222
Step 3	93.785	93.650	36.240
Step 5	58.811	59.439	27.138
WPD-RELM <i>vs.</i> WPD-SVM			
Step 1	93.651	93.829	49.526
Step 3	70.213	69.511	29.518
Step 5	44.012	44.882	21.857

3.7.3 Analysis of Deep Learning Methods

According to the results of deep learning methods in section 3.6.3, the following conclusions can be drawn:

- (a) Among all the 1-step, 3-step, and 5-step forecasting results, the EWT-GMDH model has the best results, and the variation trend of its prediction results is basically the same as that of the real value, with the highest accuracy. However, the EWT-CNN model is relatively worse. Also, it achieves the worst evaluation indices, with a MAPE of 0.3435% when 5-step ahead forecasting. This shows that convolution operation is not appropriate for time series. When the decomposition algorithms are the same, the accuracy of the GMDH is higher. The improving percentages of EWT-GMDH compared to other deep-learning models are shown in table 3.7.

TAB. 3.7 – Improving percentages of the EWT-GMDH model.

Improving percentages	P_{MAE} (%)	P_{MAPE} (%)	P_{RMSE} (%)
EWT-GMDH <i>vs.</i> EWT-CNN			
Step 1	93.363	93.319	93.750
Step 3	83.128	83.074	83.713
Step 5	80.576	80.614	80.460
EWT-GMDH <i>vs.</i> EWT-DBN			
Step 1	95.298	95.262	95.276
Step 3	88.859	88.841	88.532
Step 5	87.081	87.132	86.290

- (b) The three kinds of deep learning models show sensitivity to the number of forecasting steps. Especially for the EWT-DBN and EWT-CNN models, when the number of forecasting steps is greater than 1, the accuracy drops sharply, and the predicted curve can no longer meet the accuracy requirements. For instance, the predicted values of the 905th sample are far away from the actual power series. Conclusively, the EWT-GMDH is a better deep-learning model.

3.8 Mobile Robot Delivery Process Control Based on Power Forecasting

The delivery process control of the mobile robot is a complex system supported by several subsystems, including the navigation control subsystem, power management subsystem, and power recharging system. Power management is an important part of the mobile robot delivery process. Power management guarantees the delivery process has enough power, and determines whether to use the power recharging system according to the residual power. The power recharging system should also be optimized by intelligent or deep learning methods. The installation of the recharging station and the arrangement of recharging should be optimized.

A brief structure of different subsystems in the mobile robot delivery process is shown in figure 3.25. It can be seen that the distributed task is transmitted to each mobile robot by the wireless bridge. The navigation control and path planning system plan the delivery path for the mobile robot. The power management system should determine whether the mobile robot has enough residual power and guarantee the power voltage for the whole delivery process. To realize those functions, power voltage forecasting is the most important function of the power management system.

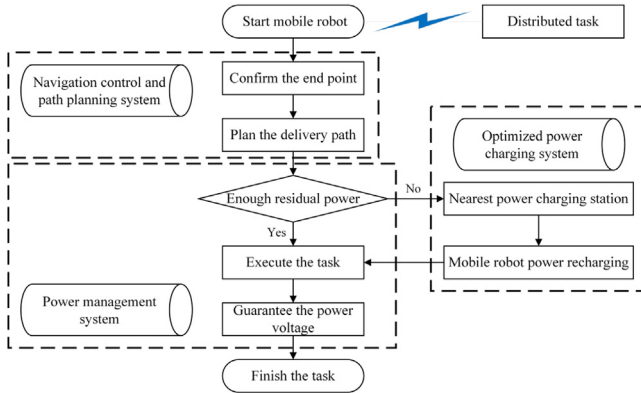


FIG. 3.25 – Structure of mobile robot delivery process control.

3.9 Conclusions

Power management and health monitoring systems are important for a mobile robot. The power time series forecasting can assist in power management effectively. In this section, the power time series forecasting for a mobile robot is introduced. Especially, the mobile robot power voltage time series modeling and forecasting methods are introduced in detail. Different statistical models, hybrid intelligent models, and hybrid deep learning models are built for power time series forecasting.

Combing the results of different methods, the following conclusions of mobile robot power time series forecasting can be drawn:

- (a) The ARIMA statistical model has sufficient stability in multi-step power voltage forecasting. However, the accuracy performance of the ARIMA method cannot meet the requirement of practical applications in the power management system. Future research should focus on how to improve the accuracy without decreasing the stability of the ARIMA model in mobile robot power time series forecasting.
- (b) Three intelligent models can forecast the mobile robot's power voltage time series effectively. Among them, the WPD-RELM is proven to be the best intelligent forecasting model for mobile robot power management systems. However, the RELM method can be affected by the forecasting steps.
- (c) Deep learning methods have better robustness because they can extract deeper information from power time series signals. Three deep learning methods have effective performance in power voltage time series forecasting. Among them, the WPD-GMDH model has the best accuracy and stability. So, the GMDH method is an effective deep learning method for mobile robot power voltage forecasting.

References

- Amini M., Karabasoglu O., Ilić M. D., Boroojeni K. G., Iyengar S. (2015) *Arima-based demand forecasting method considering probabilistic model of electric vehicles' parking lots*. Paper presented at the 2015 IEEE Power & Energy Society General Meeting.
- Amini M. H., Kargarian A., Karabasoglu O. (2016) ARIMA-based decoupled time series forecasting of electric vehicle charging demand for stochastic power system operation, *Electr. Power Syst. Res.* **140**, 378–390.
- Arvin F., Watson S., Turgut A. E., Espinosa J., Krajník T., Lennox B. (2018) Perpetual robot swarm: long-term autonomy of mobile robots using on-the-fly inductive charging, *J. Intell. Robot. Syst.* **92**(3–4), 395–412.
- de Oliveira E. M., Oliveira F. L. C. (2018) Forecasting mid-long term electric energy consumption through bagging ARIMA and exponential smoothing methods, *Energy* **144**, 776–788.
- Deng W., Zheng Q., Chen L. (2009) *Regularized extreme learning machine*. Paper presented at the 2009 IEEE symposium on computational intelligence and data mining.
- Ghelardoni L., Ghio A., Anguita D. (2013) Energy load forecasting using empirical mode decomposition and support vector regression, *IEEE Trans. Smart Grid* **4**(1), 549–556.
- Gilles J. (2013) Empirical wavelet transform, *IEEE Trans. Signal Process.* **61**(16), 3999–4010.
- Gokhale M., Khanduja D. K. (2010) Time domain signal analysis using wavelet packet decomposition approach, *Int. J. Commun. Network Syst. Sci.* **3**(03), 321–329.
- Hamza A., Ayanian N. (2017) *Forecasting battery state of charge for robot missions*. Paper presented at the Proceedings of the Symposium on Applied Computing.
- Hinton G. E., Osindero S., Teh Y.-W. (2006) A fast learning algorithm for deep belief nets. *Neural Comput.* **18**(7), 1527–1554.
- Krizhevsky A., Sutskever I., Hinton G. E. (2012) *Imagenet classification with deep convolutional neural networks*. Paper presented at the Advances in neural information processing systems.
- Liu H. (2020) *Prediction Models of Urban Air Quality in Smart Environment*. In: Smart Cities: Big Data Prediction Methods and Applications. Springer, Singapore. https://doi.org/10.1007/978-981-15-2837-8_8.

- Liu H., Chen C., Lv X., Wu X., Liu M. (2019) Deterministic wind energy forecasting: A review of intelligent predictors and auxiliary methods, *Energy Convers. Manag.* **195**, 328–345.
- Liu H., Stoll N., Junginger S., Thurow K. (2013) *An application of charging management for mobile robot transportation in laboratory environments*. Paper presented at the 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).
- Liu H., Stoll N., Junginger S., Thurow K. (2014) *A new approach to battery power tracking and predicting for mobile robot transportation using wavelet decomposition and ANFIS networks*. Paper presented at the 2014 IEEE International Conference on Robotics and Biomimetics (ROBIO).
- Madala H. R., Ivakhnenko A. G. (1994) *Inductive learning algorithms for complex systems modeling* (Vol. 368). CRC press Boca Raton.
- Nichiforov C., Stamatescu I., Făgărășan I., Stamatescu G. (2017) *Energy consumption forecasting using ARIMA and neural network models*. Paper presented at the 2017 5th International Symposium on Electrical and Electronics Engineering (ISEEE).
- Pearce M., Mutlu B., Shah J., Radwin R. (2018) Optimizing makespan and ergonomics in integrating collaborative robots into manufacturing processes, *IEEE Trans. Autom. Sci. Eng.* (99), 1–13.
- Pentzer J., Brennan S., Reichard K. (2014) *On-line estimation of vehicle motion and power model parameters for skid-steer robot energy use prediction*. Paper presented at the American Control Conference (ACC), 2014.
- Smola A. J., Schölkopf B. (2004) A tutorial on support vector regression, *Stat. Comput.* **14**(3), 199–222.
- Specht D. F. (1991) A general regression neural network, *IEEE Trans. Neural Netw.* **2**(6), 568–576.
- Sun W., Liu M. (2016) Wind speed forecasting using FEEMD echo state networks with RELM in Hebei, China, *Energy Convers. Manag.* **114**, 197–208.
- Wang J., Wang Y., Li Y. (2018) A novel hybrid strategy using three-phase feature extraction and a weighted regularized extreme learning machine for multi-step ahead wind speed prediction, *Energies* **11**(2), 321.
- Zou B., Xu X., De Koster R. (2018) Evaluating battery charging and swapping strategies in a robotic mobile fulfillment system, *Eur. J. Oper. Res.* **267**(2), 733–753.

Chapter 4

Robot Arm Time Series Predictive Control

4.1 Introduction

With the rapid development of industry, the requirements for the working accuracy of industrial robots are getting higher and higher. The robot arm is an essential component of the robot operating system. In engineering applications, the highest accuracy requirement of the robot arm is reflected in the posture accuracy (position and orientation accuracy) of the tool center point of the robot arm. The working precision of the robot will decrease gradually because of the robot arm's deterioration aging and wearing, which causes a drop in manufacturing quality and production efficiency. Therefore, it is crucial to assess the operational status of the robot arm to prevent the negative impacts of this circumstance.

At present, scholars have carried out a series of research on the data of the robot arm, and on this basis, the working condition of the robot is diagnosed and predicted. Qiao and Weiss (2017) diagnosed and predicted the health of robots from the aspects of test method modeling and algorithm development, advanced sensor development for measuring 7-D information (time, X, Y, Z, rolling, pitching, and yawing), and data analysis algorithm. P. W. Huang and Chung (2019) proposed a robot arm fault diagnosis system based on an ensemble machine learning model, which was used to estimate the Cartesian positioning deviation of the wafer-handling robot arm. Riazi *et al.* (2019) used the robot arm to study the torque required to move the arm forward and backward to identify the typical faults of the system. Y.-C. Huang *et al.* (2019) used feature extraction to extract synthetic position signals (x, y), and used Generalized Regression Neural Network (GRNN) and logistic regression models for data mining to predict the lifetime of the Wafer Handling Robot Arm (WHRA). Lee *et al.* (2019) proposed a Sequence to Sequence (Seq2Seq) model which uses time series data to transform between artificial neural network models to forecast the faults of industrial robots. This method uses the joint current and angle data that can be measured by the robot itself for fault diagnosis and does not need additional sensors. Hu *et al.* (2019) used the acoustic feature information of the robot's dynamic arm to make intelligent fault predictions for the robot.

In this chapter, a time series prediction model of the robotic arm is developed. The model predicts the next parameter from the historical parameters of the robotic arm. Building a robotic arm time prediction model has the following two advantages.

- (a) The prediction model helps to monitor the actual working status of the robotic arm. The deviation of the actual value from the predicted value is used to determine whether the current robotic arm is working normally and to provide feedback on abnormal conditions in advance.
- (b) The predicted value can give the possible values of the future robot arm coordinates in advance. Based on this, deviations are calculated so that the system can correct deviations in advance and improve the working accuracy of the robotic arm.

First, common processing parties of robot arm time series are summarized and analyzed. Then, statistical models, intelligent models, and deep learning models for robot arm prediction are introduced and the advantages and disadvantages of different prediction algorithms are compared. Finally, the application of robot arm time series prediction algorithms in industrial robots is described. Figure 4.1 illustrates the overall structure of the robot arm prediction model.

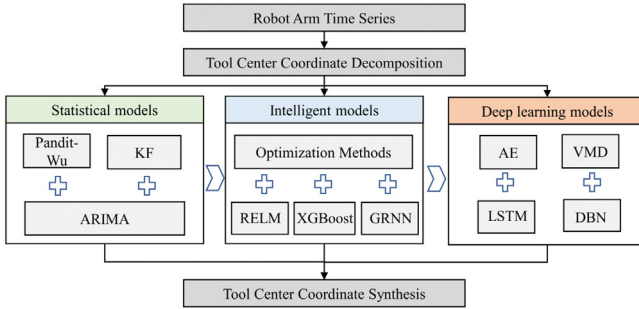


FIG. 4.1 – Research flow and corresponding sections of chapter 4.

4.2 Robot Arm Time Series Measurement

To achieve accurate control of the robot arm, the position, velocity, acceleration, and torque of each node of the robot arm need to be accurately measured. Currently, scholars have focused on two aspects of robot arm measurement, developing more advanced sensing devices and processing the data measured by the sensing devices.

The sensing devices of the robot arm can be divided into mechanical sensors, electromagnetic sensors, and optical sensors. Mechanical sensors are gyroscopes and inertial acceleration sensors. Optical sensors have the advantage of high accuracy and the disadvantage of high manufacturing costs. Presently, many scholars are working on more advanced sensing devices. Cheng and Oelmann (2009) introduced the measurement and analysis of robot arm joint angles based on microelectromechanical

systems with two-axis accelerometers and single-axis gyroscopes. Sareh *et al.* (2015) presented a position and orientation sensing system for a soft robot arm with macroscopic bending and stretching sensors, which consisted of optical fibers. Jhang *et al.* (2017) designed a multi-sensor glove controller for controlling an industrially operated six-axis robot arm. Georgopoulou *et al.* (2021) investigated silicone elastomer-based piezoresistive sensors to monitor the position of a robot arm.

The processing of sensor acquisition data includes the fusion of different sensor acquisition data as well as the processing and mining of sensor acquisition data. Jassemi-Zargani and Neculescu (2002) used the extended Kalman filter to fuse multiple sensors to improve the measurement accuracy. Pradeep *et al.* (2014) corrected the sensor measurement data to reduce the uncertainty of sensor measurements. Cantelli *et al.* (2015) used the Kalman filter algorithm to fuse measurements from a three-axis gyroscope and accelerometer to estimate joint angles. Widmaier *et al.* (2016) proposed a method for estimating joint angles and positions using images of robot arm poses. Qiao (2019) proposed a vision-based 6-DOF measurement system that can measure the dynamic motion of a robot in real-time.

4.3 Robot Arm Time Series Uncertainty Analysis

The uncertainty of the robot arm time series will directly affect the prediction accuracy of the model. Therefore, it is necessary to analyze the uncertainty of the robot arm time series. The uncertainty of the robot arm time series is affected by two factors: the environment and the measurement uncertainty.

- (a) The environment can negatively affect the robot arm. The robot arm works mainly through a series of sensing devices, and the sensing devices usually have high requirements for the working environment. A poor working environment will greatly reduce the accuracy of the sensing devices, which can increase the uncertainty of the whole system.
- (b) The collected data sets are usually somewhat noisy due to interference with the sensor devices as they measure and collect data. In addition, as the robot arm is used, the joints wear and age, and the relative positions between the arms change, which increases the uncertainty of the collected data.

The uncertainty of the robot arm sequence cannot be fundamentally eliminated. Therefore, it is crucial to improve the robustness of the robot arm prediction model. If the uncertainty in the sequence is within the allowed range, a robust prediction model can still achieve high accuracy.

4.4 Robot Arm Time Series Statistical Forecasting Method

Statistical prediction methods are still the most frequent and widely used effective prediction method in time series modeling. In this section, two different statistical prediction methods for robot arm time series are described, including the Pandit–Wu prediction algorithm and the Kalman filter (KF) prediction algorithm.

4.4.1 Pandit–Wu Forecasting Algorithm

The Pandit–Wu modeling method is a system modeling method proposed by Pandit and S. M. Wu on the basis of the Box–Jenkins method (Pandit *et al.*, 1984). The strategy is utilizing the Autoregressive Moving Average (ARMA) model to estimate the time series X_t .

(1) Algorithm principle

The parameters n of the $ARMA(2n, 2n - 1)$ is increased until the sum of residual variances did not decrease significantly. Figure 4.2 illustrates the process of the Pandit–Wu algorithm.

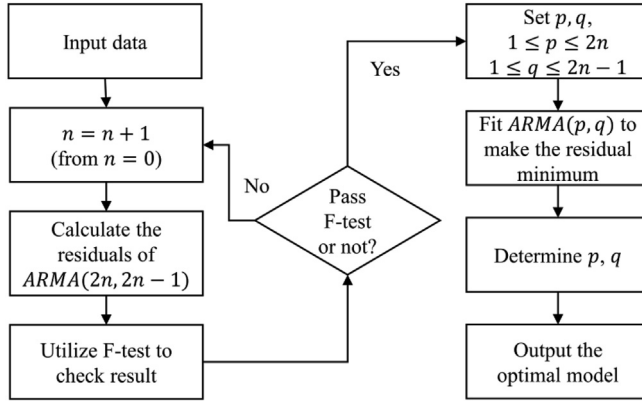


FIG. 4.2 – Pandit–Wu forecasting algorithm flow.

(2) Modeling flow

The specific modeling process consists of data processing, parameter estimation, and prediction.

- Data processing. The robot arm sequence is divided into 6 subsequences, X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle. Then, the sequences become smooth by differencing and zero-averaging.
- Parameters estimating. The order and parameters of the ARMA model are determined. The order of the ARMA model is obtained by searching for the optimal parameters. The maximum search range is determined by the F-test.
- Prediction. The prediction model built by the Pandit–Wu algorithm is tested on the test set.

(3) Data processing

The original robot arm time series is composed of the 6-dimensional tool center position coordinates of the robot arm. The data contains 7909 sets of data with a sampling frequency of 125. The data shows that this robot's arm motion is periodic. Therefore, one cycle of data (from 1801 to 5100) is taken out for the study. The first 2310 sets are used as the training set and the last 990 sets are used as the test set.

The Pandit–Wu prediction model is based on the ARMA model, which is not suitable for the prediction of multidimensional data. Therefore, the original tool center position time series was divided into six subseries. Figure 4.3 shows the subseries obtained from the decomposition. The Pandit–Wu forecasting algorithm requires the data to be stationary. Therefore, the ADF test is needed to test the smoothness of the series. If the test is not passed, the original series needs to be processed to make it smooth.

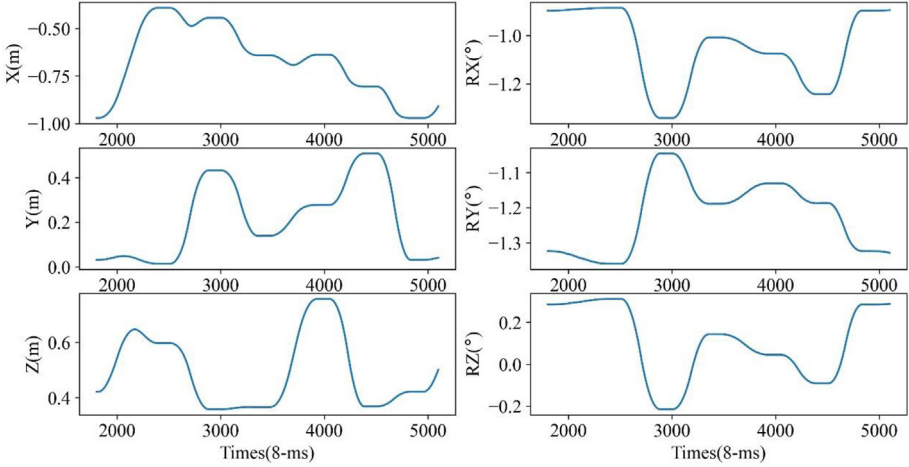


FIG. 4.3 – The subseries of the robot arm.

(4) Parameter estimation

Take the X-position series as an example. First, determine the maximum search order of p in the $\text{ARMA}(2p, 2p - 1)$. Then, starting from $p = 1$, gradually increase the order of the model and use the $\text{ARMA}(2p, 2p - 1)$ to fit. Finally, determine the most suitable parameters for the model.

Depending on the results of the fitting, p is taken as 2 or 3. According to the F-test, there is no significant difference in the variance when $p = 2$ and $p = 3$. To make the model simpler, p is taken as two. Then, use the $\text{ARMA}(m, n)$ ($1 \leq m \leq 4$, $1 \leq n \leq 3$) to fit the data, and take the minimum residual sum as the optimization objective to find the most suitable parameter. According to the result of calculation, the residual sum is the smallest when $m = 4$ and $n = 2$. Therefore, the optimal model corresponding to the X-position series is the $\text{ARMA}(4, 2)$. The equation of the model is shown in equation (4.1).

$$X_t = 0.446X_{t-1} + 1.500X_{t-2} - 0.350X_{t-3} - 0.596X_{t-4} + \varepsilon_t + 1.012\varepsilon_{t-1} - 0.056\varepsilon_{t-2} \quad (4.1)$$

where ε_t obeys Gaussian distribution $\varepsilon_t \sim (0, 2e - 7)$.

(5) *Forecasting*

Models suitable for each subsequence were obtained separately by parameter estimation. To test the prediction accuracy of the Pandit–Wu algorithm, each model is validated on the test set. Figure 4.4 shows the predicted and actual values.

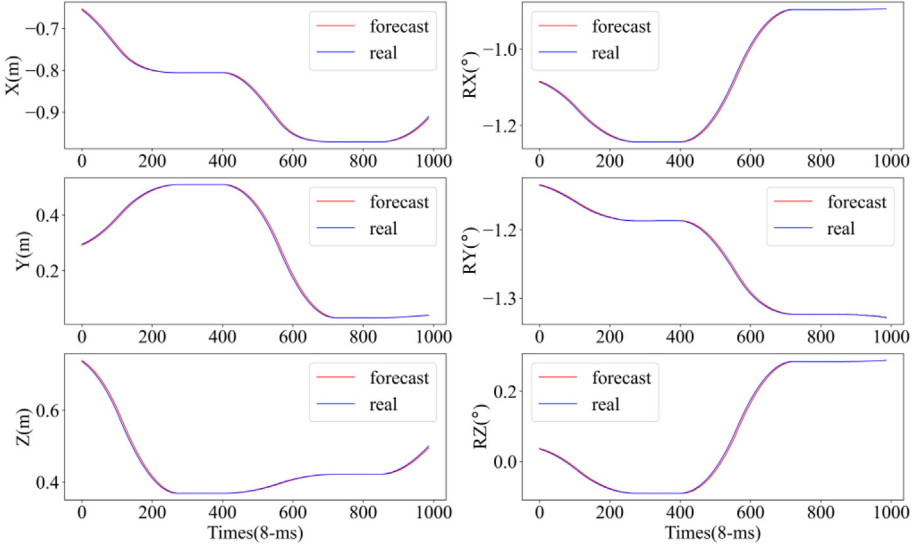


FIG. 4.4 – The ground truth and predicted value of the Pandit–Wu model.

The Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE) are used to evaluate the model prediction results. Table 4.1 describes the results of the evaluation. The results show a large difference in the prediction accuracy of the model for each series.

TAB. 4.1 – The evaluation indices of the Pandit–Wu prediction model.

	X-position series	Y-position series	Z-position series
MAE (mm)	1.931	3.568	2.536
RMSE (mm)	2.710	5.434	4.093
MAPE (%)	0.234	1.869	0.510
	RX-angle series	RY-angle series	RZ-angle series
MAE (10^{-3}°)	2.590	1.037	2.593
RMSE (10^{-3}°)	3.896	1.548	4.089
MAPE (%)	0.238	0.084	21.255

4.4.2 *KF-ARMA Forecasting Algorithm*

The KF is a kind of state estimation proposed by Kalman (1960). It provides an optimal estimation of the real-time state of a finite-dimensional stochastic system. The KF uses statistical information of noise and system status and takes minimizing

mean square error as the optimization goal to give an optimal estimation of the original input signal. In this section, the KF is used to enhance the anti-interference ability of the Autoregressive Moving Average (ARMA) model when the original data have noise signals.

(1) *Algorithm principle*

The KF includes two parts, the prediction part, and the state update part, as shown in figure 4.5.

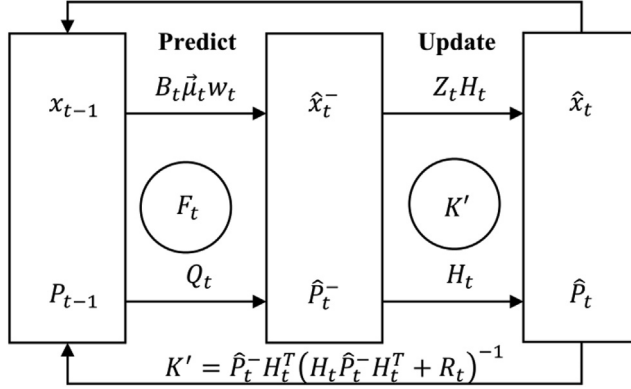


FIG. 4.5 – The construct of KF.

The prediction part describes the relationship between the optimal estimator $\hat{x}_{t-1}$ at the previous moment and the prior estimate $\hat{x}_t^-$, and it is defined as follows (Welch and Bishop, 1995):

$$\hat{x}_t^- = F_t \hat{x}_{t-1} + B_t \bar{\mu}_t + w_t \quad (4.2)$$

$$\hat{P}_t^- = F_t \hat{P}_{t-1} F_t^T + Q_t \quad (4.3)$$

where $\bar{\mu}_t$ is determined in the physical environment, w_t is the interference value determined in the physical environment, F_t and B_t are respectively the weight value corresponding to item weight, $\hat{P}_t$ is the co-matrix of the prior estimation part, and Q_t is the process noise.

Through the state update part, the result of the posterior estimation is obtained as the final result at the current moment (Welch and Bishop, 1995).

$$\hat{x}_t = \hat{x}_t^- + K' (\bar{z}_t - H_t \hat{x}_t^-) \quad (4.4)$$

$$\hat{P}_t = \hat{P}_t^- - K' H_t \hat{P}_t^- \quad (4.5)$$

$$K' = \hat{P}_t^- H_t^T (H_t \hat{P}_t^- H_t^T + R_t)^{-1} \quad (4.6)$$

where $\hat{x}_t$ is the result of Kalman prediction at time t , $\hat{P}_t$ represents the corresponding covariance matrix, and K' is the Kalman gain at time t .

(2) Modeling flow

The specific modeling process consists of data processing, parameter estimation, and prediction.

- (a) Data processing. The original tool center position data is decomposed. Then noise is added to the decomposed subsequence. Then KF is utilized to process the sub-series with noise. Then check the stationarity of the sub-series.
- (b) Parameter estimation. The Pandit–Wu algorithm and Akaike Information Criterion (AIC) are used to obtain the order p and q in the ARMA(p, q) model. Then the remaining parameters are estimated.
- (c) Prediction. The predicted value of the model is tested and compared with the different models.

(3) Data preprocessing

First, the original tool center position sequence is divided into 6 sub-series, X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle. To reduce the time required to calculate model parameters, the sampling interval is set to 10. To test the robustness of the model, the artificial noise signal is added based on the original series. The noise obeys the Gaussian distribution $\delta \sim (0, 0.01)$. Then, the series with noise is processed by KF. Figure 4.6 shows that the series with KF is smoother.

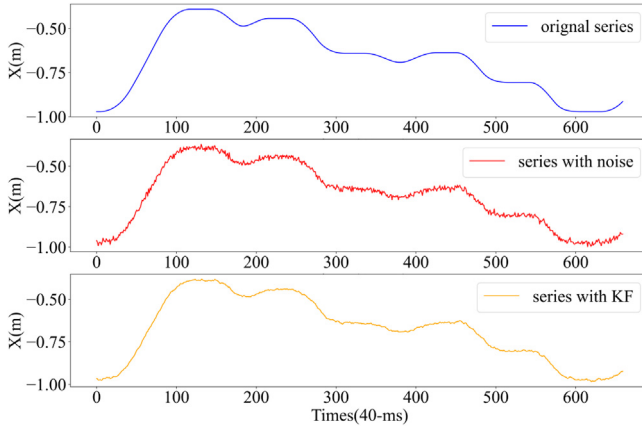


FIG. 4.6 – The result of the series with KF.

Then the stationarity test is carried out on the three series respectively. According to the ADF test, the three series are all non-stationary. It is necessary to use the difference to make the series stationary, as shown in figure 4.7. The result of figure 4.7 shows that noise affects the result of difference. The first-order difference processing is performed on the three. Through processing, three series are stationary.

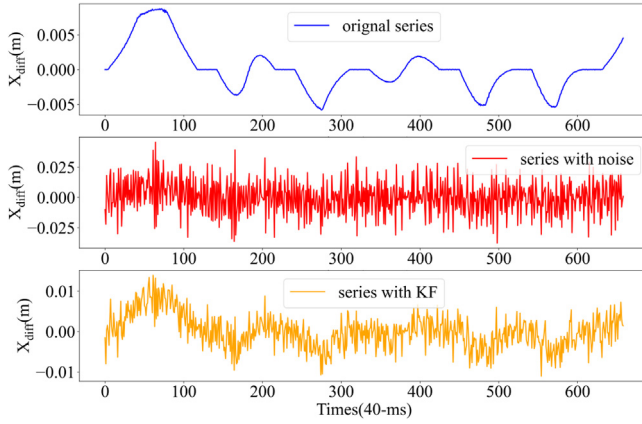


FIG. 4.7 – The result of difference for the three series.

(4) Parameter determination

Take the X-position series processed by the KF as an example. The Pandit–Wu algorithm is used to determine the maximum search order of p in the $\text{ARMA}(2p, 2p - 1)$. The residual sum of squares of the $\text{ARMA}(2p, 2p - 1)$ has no longer significantly decreased when $p = 3$. Then the AIC criterion is used to determine the parameters m and n in the $\text{ARMA}(m, n)$. The AIC is defined as equation (4.7) (Burnham and Anderson, 2004).

$$\text{AIC} = 2k - 2\ln(L) \quad (4.7)$$

where L denotes the value of the maximum likelihood function in the model, and k is the number of unknown parameters. Figure 4.8 illustrates the AIC value of the $\text{ARMA}(m, n)$.

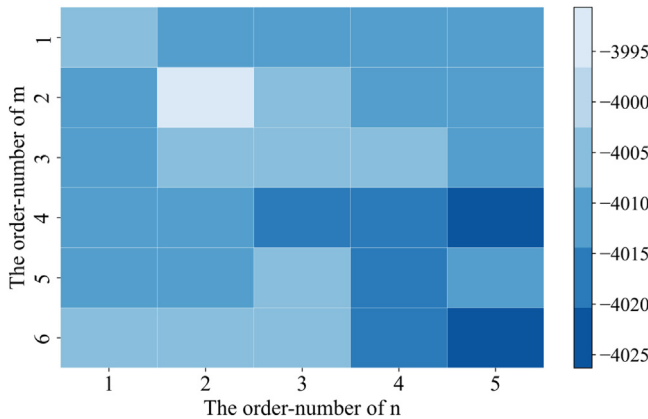
FIG. 4.8 – The AIC of $\text{ARMA}(m, n)$.

Figure 4.8 indicates that the AIC of the model is the smallest when $m = 6$ and $n = 5$. The model is shown in equation (4.8).

$$D_t = 0.579D_{t-1} - 0.448D_{t-2} + 0.411D_{t-3} - 0.469D_{t-4} + 0.936D_{t-5} + 0.122D_{t-6} + \varepsilon_t + 0.638\varepsilon_{t-1} + 0.659\varepsilon_{t-2} - 0.137\varepsilon_{t-3} - 0.173\varepsilon_{t-4} - 0.770\varepsilon_{t-5} \quad (4.8)$$

where ε_t obeys Gaussian distribution $\varepsilon_t \sim (0, 9.15e-06)$.

The model without processing by the KF is shown in equation (4.9).

$$D_t = 1.211D_{t-1} - 0.126D_{t-2} + 0.117D_{t-3} - 0.214D_{t-4} + 0.936D_{t-5} + \varepsilon_t + 0.638\varepsilon_{t-1} + 0.659\varepsilon_{t-2} \quad (4.9)$$

where ε_t obeys Gaussian distribution $\varepsilon_t \sim (0, 1.35e-4)$.

(5) Forecasting

The model obtained above is tested on the test set. The model is used to predict the differential values and then the sequence is recovered. Figure 4.9 shows that the model using KF has a smaller prediction error. This indicates that adding KF to the model can improve the robustness of the model.

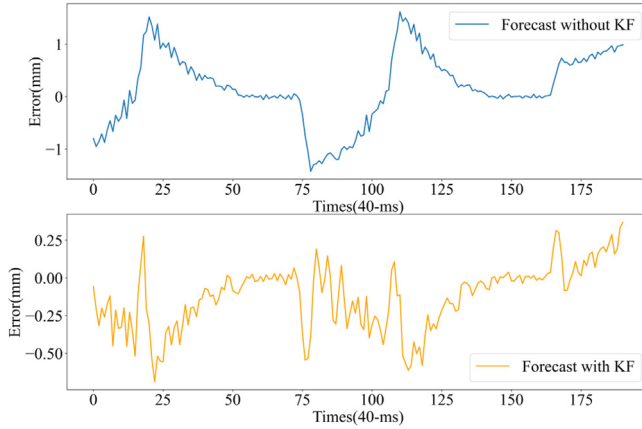


FIG. 4.9 – Comparison of prediction errors.

To further compare the performance of different models, the MAPE, RMSE, and MAE are obtained, as shown in table 4.2. Table 4.2 demonstrates the forecasting error of the model is obviously reduced with KF, which indicates that KF can improve the robustness of the model.

TAB. 4.2 – The evaluation indices of the ARIMA and KF-ARIMA.

	ARIMA	KF-ARIMA
MAE (10^{-4} m)	5.212	1.807
RMSE (10^{-4} m)	6.840	2.437
MAPE (%)	0.061	0.021

4.5 Robot Arm Time Series Intelligent Forecasting Method

When the sample size is small, the forecasting performance of the intelligent algorithm is often better than the general statistical model. In this section, three intelligent prediction algorithms are introduced, Regularized Extreme Learning Machine (RELM), Extreme Gradient Boosting (XGBoost), and GRNN.

4.5.1 RELM Forecasting Algorithm

Extreme Learning Machine (ELM) is a feed-forward neural network learning algorithm with only one hidden layer. The large number of parameters in the neural network makes the network training slow. However, during the parameter calculation of the Extreme Learning Machine, the input weights and biases of the network are kept constant after random initialization. Only the weights between the output and hidden layers are computed, making the computation of the entire network very fast (G.-B. Huang *et al.*, 2011).

However, during the training process, ELM is prone to overfitting. To solve the overfitting problem in model prediction, Zhang and Zhang (2016) proposed RELM for the sparsity of the matrix, a regularization term is added to the objective function of the original ELM model to obtain RELM. The L2 criterion regularization is used to obtain the parameter solution with low complexity of the final model by solving the minimum of the regularized objective function.

(1) Algorithm principle

The main idea is to solve the weight $\hat{\beta}$ between the hidden and output layer according to the least squares principle. Figure 4.10 shows the structure of the ELM.

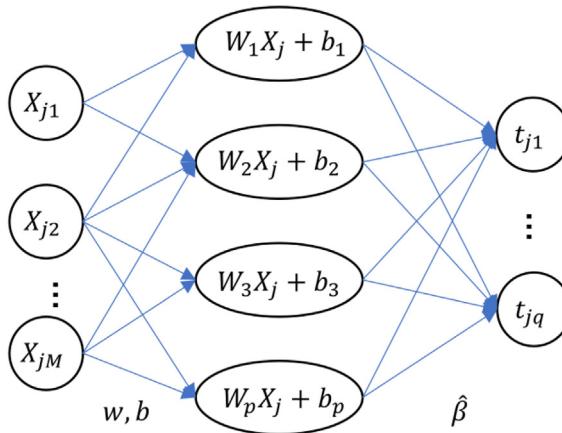


FIG. 4.10 – The structure of ELM.

For an ELM network, given the input value $\{\mathbf{x}_j, j = 1, 2, 3, \dots, N\}$ and output value $\{t_j, j = 1, 2, 3, \dots, N\}$ of N samples.

For each sample, if x has m dimensions

$$\mathbf{x}_j = [x_{j1}, x_{j2}, x_{j3}, \dots, x_{jM}]^T \in R^M \quad (4.10)$$

For the network with p hidden layer units and q output layer units, the mathematical expression of its input value $\mathbf{x}_j$ and output value t_j as (G.-B. Huang *et al.*, 2011)

$$t_j = \sum_{i=1}^p \beta_i g(\mathbf{w}_i \bullet \mathbf{x}_j + \mathbf{b}_i), j = 1, 2, 3, \dots, N \quad (4.11)$$

where $\beta_i = [\beta_{i1}, \beta_{i2}, \beta_{i3}, \dots, \beta_{iq}]^T$ denotes the vector between the i -th hidden layer and the output layers, $\mathbf{w}_i$ denotes the weight between the input layer and the i -th hidden layer, $\mathbf{b}_i$ denotes the bias of the i -th node.

The objective of ELM is to decrease the difference between the predicted and actual values.

$$E = \min \sum_{j=1}^N \left(\sum_{i=1}^p \beta_i g(\mathbf{w}_i \bullet \mathbf{x}_j + \mathbf{b}_i) - t_j \right)^2 \quad (4.12)$$

To prevent the ELM from overfitting, a regular term $\frac{C}{2} \beta_i^2$ is added to the objective function (Zhang and Zhang, 2016).

$$E = \min \left\{ \sum_{j=1}^N \left(\sum_{i=1}^p \beta_i g(\mathbf{w}_i \bullet \mathbf{x}_j + \mathbf{b}_i) - t_j \right)^2 + \frac{C}{2} \beta_i^2 \right\} \quad (4.13)$$

where C denotes the penalty term coefficient, which needs to be set manually.

(2) Modeling flow

The modeling process consists of data processing, training, and prediction.

- (a) Data preprocessing. The original sequence is divided into six sub-series, which are X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle. Then, the data set is converted and divided into forms.
- (b) Training. The parameters of the RELM network are trained. However, the neuron number and penalty coefficient in the first layer of RELM need to be manually set. These two parameters can be selected from the empirical value and optimized by the optimization algorithm.
- (c) Forecasting. The effect of the trained RELM is tested on the test set.

(3) Data processing

Due to the motion of the robot arm is periodic, only part of the source data is intercepted for experiments. Figure 4.11 shows the original data of the robot arm position.

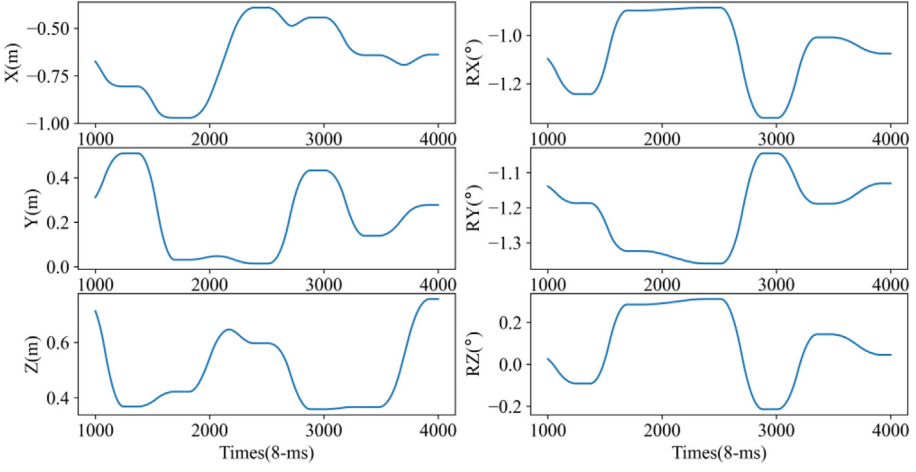


FIG. 4.11 – The original data of robot arm position.

To meet the needs of the model, it is necessary to build a connection between the position sequence $\{p_i, p_{i+1}, p_{i+2}, p_{i+3}, \dots, p_{i+N}\}$ of the first $N+1$ moments and the sequence $\{p_{i+N+1}, p_{i+N+2}, p_{i+N+3}, \dots, p_{i+N+M}\}$ of the last M moments. The original data format is 3000×1 . Here we set $N = 4$, $M = 1$ to process the data. That means using the position sequence of the first 5 moments to predict the position of the next moment. After processing, the obtained data is transformed into 2995×6 . Then the processed datasets are divided. The former 2095 sets are used for model training, and the last 900 sets are used for testing.

(4) Training

For the ELM without a regular term, the weight between the hidden layer and the output layer can be defined as (G.-B. Huang *et al.*, 2011):

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \quad (4.14)$$

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^\dagger \mathbf{y} \quad (4.15)$$

where $\mathbf{H}^\dagger$ is the pseudo-inverse matrix of $\mathbf{H}$.

For the RELM, the $\hat{\boldsymbol{\beta}}$ can be solved as shown in equation (4.16) (Zhang and Zhang, 2016).

$$\hat{\boldsymbol{\beta}} = \begin{cases} (\mathbf{H}^T \mathbf{H} + \mathbf{C}^2)^{-1} \mathbf{H}^T \mathbf{T} & N \leq M \\ \mathbf{H}^T (\mathbf{H} \mathbf{H}^T + \mathbf{C}^2 \mathbf{I})^{-1} \mathbf{T} & N > M \end{cases} \quad (4.16)$$

(5) Forecasting

Since the weight $\mathbf{w}$ is initialized randomly, the solved weights $\hat{\boldsymbol{\beta}}$ will be different. Therefore, it is important to ensure that the $\mathbf{w}$ is consistent during training and

testing. The performance of the ELM on the test set is shown in figure 4.12. Figure 4.12 indicates that the unregularized extreme learning machine performs well on the test set, and the prediction error is close to zero. The performance of the ELM with regular terms is shown in figure 4.13. Figure 4.13 indicates that the error of ELM with the regular term has increased.

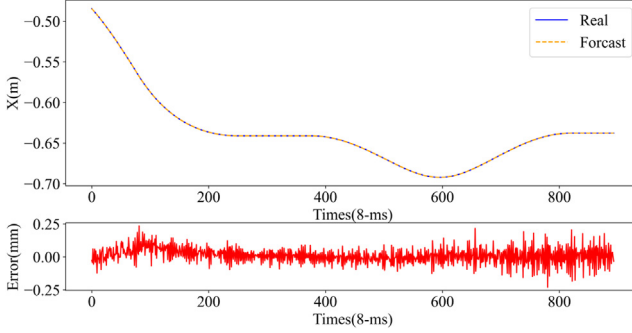


FIG. 4.12 – The true values and predicted values of the ELM.

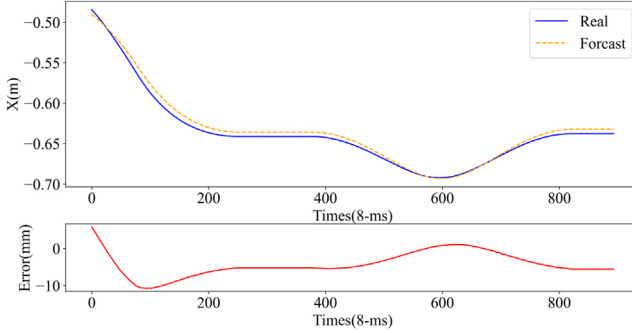


FIG. 4.13 – The true values and predicted value of the RELM.

The penalty term in the above experiment is manually set. To further improve the prediction performance of the model, an optimization algorithm is used to optimize the penalty term in the regular term. The Particle Swarm Optimization (PSO) is adopted to optimize the penalty coefficient C . Then, the RELM is tested again, as shown in figure 4.14. Figure 4.14 shows that the error of the optimized model decreases.

Table 4.3 shows that the predicted error of the RELM decreases after optimization. It indicates that optimizing the regular term coefficient C by the PSO is effective in improving the model accuracy. However, the performance of ELM is better than RELM. It can be seen from the above experiments that the RELM may not have better performance, especially when the variations of the data are simple.

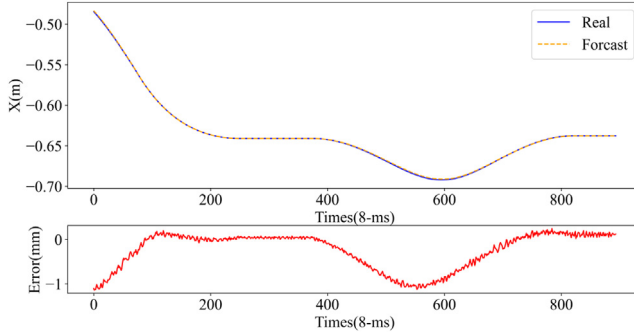


FIG. 4.14 – The predicted value of the RELM optimized by the PSO.

TAB. 4.3 – The evaluation indices of the different ELM models.

	ELM	RELM	PSO-RELM
MAE (10^{-4} m)	1.271	5.140	3.203
RMSE (10^{-4} m)	1.431	5.883	4.764
MAPE (%)	0.020	0.079	0.051

4.5.2 XGBoost Forecasting Algorithm

The XGBoost is proposed by Dr. Chen (T. Chen *et al.*, 2015). The model utilizes multiple decision tree models to learn and synthesize the results obtained from each tree model. Based on the decision tree, a regular term is set for the number of leaf nodes, which can effectively prevent the model from overfitting (T. Chen *et al.*, 2015).

(1) Algorithm principle

Giving a data set $D = \{(\mathbf{x}_i, y_i) \mid i = 1, 2, 3, \dots, N\}$, the XGBoost model obtains K decision tree models by training. The model is defined as (T. Chen *et al.*, 2015):

$$\tilde{y} = \sum_{k=1}^K f_k(\mathbf{x}) \quad (4.17)$$

where $f_k(\mathbf{x})$ denotes tree models learned from the data set. Figure 4.15 shows the flow of building the model.

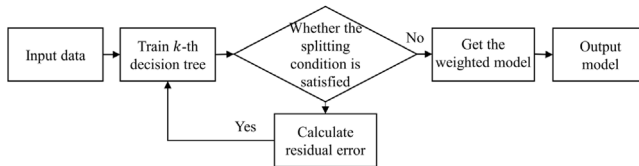


FIG. 4.15 – The flow of the XGBoost algorithm.

The whole process is aimed at minimizing the objective function, which is defined as (T. Chen *et al.*, 2015):

$$L = \sum_{i=1}^N \text{loss}(y_i, \tilde{y}_i) + \gamma T + \frac{1}{2} \lambda \|\mathbf{w}\|^2 \quad (4.18)$$

where the $\text{loss}(y_i, \tilde{y}_i)$ denotes the squares of the error between the true value and the predicted value of the sample, T denotes the number of leaf nodes, w denotes the score of leaf nodes.

The error function can change through the second-order Taylor expansion. The objective function of k -th tree model can be defined as (T. Chen *et al.*, 2015):

$$L^{(k)} = \sum_{i=1}^N \left[l(y_i, \hat{y}_i^{(k-1)} + f_k(x_i)) \right] + \gamma T + \frac{1}{2} \lambda \|\mathbf{w}\|^2 \quad (4.19)$$

Through expanding the above equation at $\tilde{y}_i^{t-1}$, the equation is rewritten as (T. Chen *et al.*, 2015):

$$L^{(k)} = \sum_{i=1}^N \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_k(\mathbf{x}_i) + \frac{1}{2} h_i f_k^2(\mathbf{x}_i) \right] + \gamma T + \frac{1}{2} \lambda \|\mathbf{w}\|^2 \quad (4.20)$$

In equation (4.20), $g_i = \partial_{\hat{y}^{(k-1)}} l(y_i, \hat{y}^{(k-1)})$, $h_i = \partial_{\hat{y}^{(k-1)}}^2 l(y_i, \hat{y}^{(k-1)})$, and $f_k(\mathbf{x}_i)$ denotes the score that ultimately belongs to that leaf node j . Finally, the $\tilde{L}^{(k)}$ can be rewritten as (T. Chen *et al.*, 2015):

$$\tilde{L}^{(k)} = \sum_{j=1}^T \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \quad (4.21)$$

where the I_j is the sample set that defines each leaf node j .

(2) Model flow

The modeling process consists of data processing, parameter estimation, and prediction.

- (a) Data processing. The data is transformed into a form that matches the model input. Then, the data is divided.
- (b) Parameter estimation. The corresponding data is brought into the model, the model parameters are calculated, and the parameters that need to be manually set are optimized.
- (c) Prediction. The results of the test set are predicted and compared with the actual results.

(3) Data processing

Since the XGBoost model is based on the decision tree model, it is susceptible to the maximum $y_{\max}$ and minimum values $y_{\min}$ of the data. The predicted value of the model will be compressed within $[y_{\min}, y_{\max}]$. Therefore, the test set should contain the maximum and minimum. The former 2300 sets are used for model training and the last 900 sets are used for testing as shown in figure 4.16.

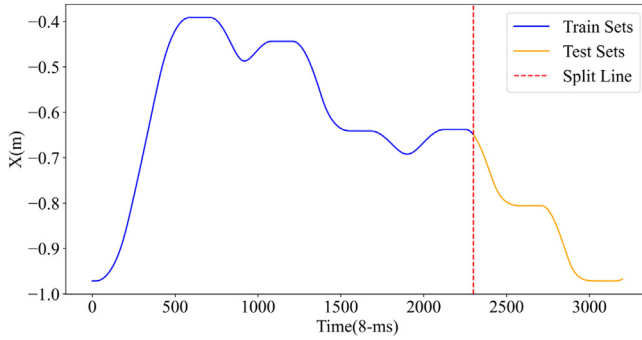


FIG. 4.16 – The split of the train set and test set for XGBoost.

(4) Training and forecasting

The XGBoost model has many uncertain parameters, such as learning rate, maximum depth, regular term coefficients, etc. To decrease the error of the model, it is important to preset the value range and value method of the parameters. The Optuna package is then used to optimize the model parameters. Through the optimization process, it is found that there are six parameters that mainly affect the model performance as shown in figure 4.17.

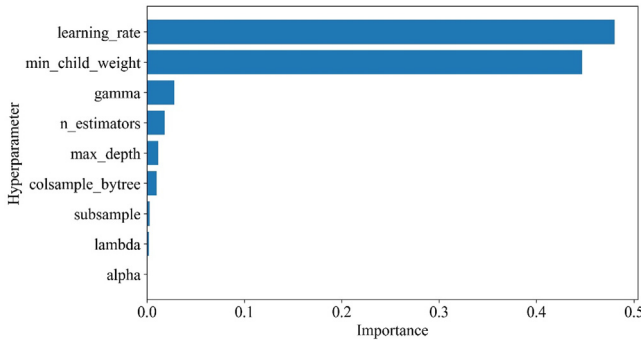


FIG. 4.17 – The importance of different parameters.

The optimization process curve is obtained as shown in figure 4.18 with further optimization. The optimization experiment is carried out 100 times and the minimum error of different parameter values is shown in figure 4.18. The model parameters are set according to the above optimization results. Figure 4.19 shows the model effect on the test set. It can be seen from figure 4.19 that the error model is large for some points, especially in the middle part.

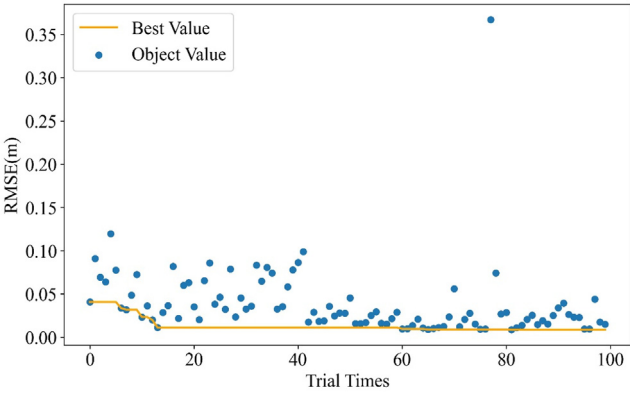


FIG. 4.18 – The process of optimization.

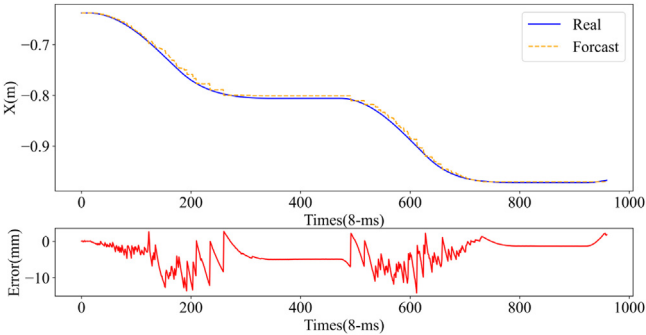


FIG. 4.19 – The true value and predicted value of the XGBoost.

To test the performance of XGBoost, several boosting models are compared, such as the Adaptive Boosting (AdaBoost), the XGBoost, and the Gradient Boosting Decision Tree (GBDT). The indicators MAE, RMSE, MAPE, and Mean Error (ME) of each boosting model are compared. The ME is defined as

$$\text{ME} = \max\{|y_i - \hat{y}_i|\} \quad i = 1, 2, 3, \dots, N \tag{4.22}$$

where y_i denotes the true value of the sample, $\hat{y}_i$ denotes the forecasting value of the sample. The results are shown in table 4.4. It is shown that the performance of XGBoost is not the best. It may be due to the overfitting ability of XGBoost.

TAB. 4.4 – The performance of the boosting model.

	XGBoost	AdaBoost	GBDT
MAE (mm)	6.145	9.748	3.056
RMSE (mm)	0.616	1.558	0.218
MAPE (%)	0.756	1.105	0.376
ME (m)	0.022	0.034	0.019

4.5.3 GRNN Forecasting Algorithm

GRNN is based on nonparametric regression. It takes the distribution of the training sample data as the posterior condition, on which the parameter estimation is performed. The output value of the network is obtained according to the maximum probability criterion. It has good nonlinear approximation performance and its training process is simpler because of the Radial Basis Function (RBF) (Specht, 1991).

(1) Algorithm principle

The architecture of the GRNN consists of four layers, which is similar to that of RBF. Figure 4.20 shows the details of the architecture.

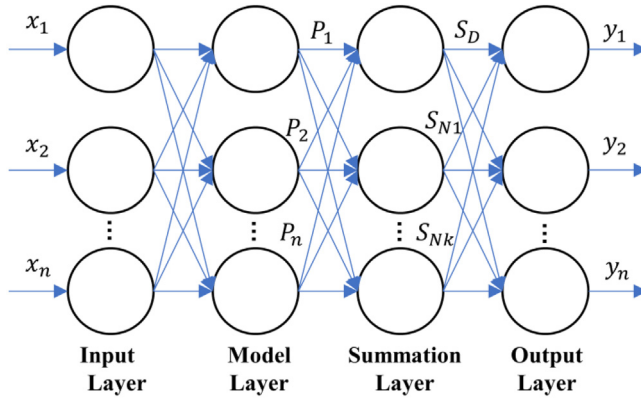


FIG. 4.20 – The architecture of the GRNN.

The input variables $\mathbf{x} = [x_1, x_2, x_3, \dots, x_n]^T$ are directly passed to the model layer through the input layer. The model layer stores the parameters of various training samples. Its transfer function is defined as (Specht, 1991):

$$P_i = \exp \left[-\frac{(\mathbf{x} - \mathbf{x}_i)^T (\mathbf{x} - \mathbf{x}_i)}{2\sigma^2} \right] \quad i = 1, 2, 3, \dots, n \quad (4.23)$$

where $\mathbf{x}$ denotes the input value, $\mathbf{x}_i$ denotes the i -th variable in the training sample, which is the learning sample stored in the i -th neuron of the model layer, P_i denotes the output result of each neuron, and it will be passed to the summation layer.

Two different neurons S_D and S_{Nj} are adopted in the summation layer. S_D is the direct accumulation of all weights in the model layer as shown in equation (4.24). S_{Nj} is the accumulation of weighted output values that correspond to the output value $\mathbf{y} = [y_1, y_2, y_3, \dots, y_n]^T$ in the sample, as shown in equation (4.25) (Specht, 1991).

$$S_D = \sum_{i=1}^n P_i \quad (4.24)$$

$$S_{Nj} = \sum_{i=1}^n y_{ij} P_i \quad i = 1, 2, 3, \dots, n \quad (4.25)$$

where y_{ij} denotes the j -th value in the i -th training sample.

The output layer utilizes that weighted sum to divide the sum of the weighted output values S_{Nj} obtained by the summation layer and the sum of weights S_D to obtain the output y_i , as shown in equation (4.26) (Specht, 1991).

$$y_i = \frac{S_{Nj}}{S_D} \quad j = 1, 2, 3, \dots, k \quad (4.26)$$

(2) Modeling flow

The modeling process consists of data processing, training model, and prediction.

- (a) Data processing. The data is transformed into a form that matches the model input. Then, the data is divided.
- (b) Training. The data corresponding to the training sample is brought into the model to obtain a model that can be used for prediction.
- (c) Prediction. The performance of the model is inspected on the test set.

(3) Data processing

The tool center position data in the data set is decomposed into 6 sub-series, which are X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle, as shown in figure 4.21. Figure 4.21 shows that the movement of the robot arm has a certain periodicity. Because the motion of the robot arm is periodic, the samples in the test set can easily match the very similar samples in the training set. Therefore, only part of the sub-series is utilized in the dataset to get the model. The position time series $[P_i, P_{i+1}, P_{i+2}, P_{i+3}, P_{i+4}]$ of the previous 5 moments is used to predict the position data P_{i+5} of the next moment.

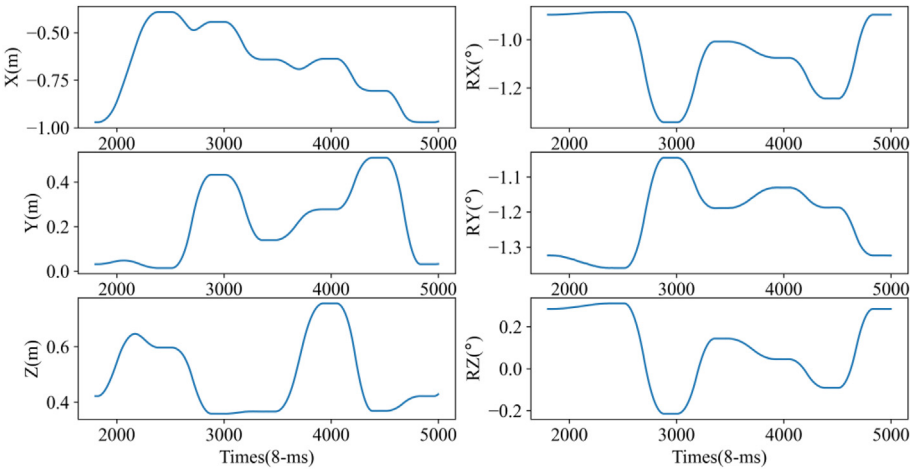


FIG. 4.21 – The original data of six sub-series.

(4) Training and forecasting

The training samples are passed into the network to build the network. After that, set the smoothing coefficient to 0.01. Then the data belonging to the test set is input into the model to inspect the performance of the model. The results of the test are shown in figure 4.22. As can be seen from figure 4.22, the performance of the model is bad for some sample data.

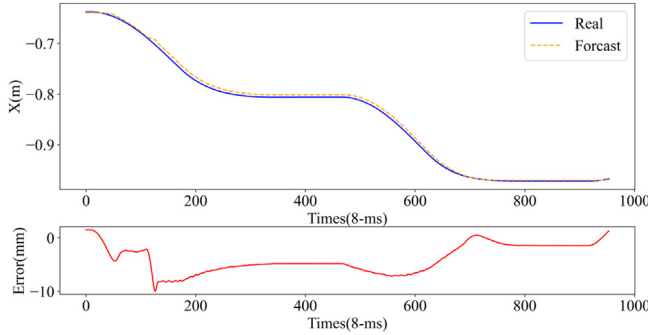


FIG. 4.22 – The true value and predicted value of the GRNN.

It is known that the smoothing coefficient has a profound influence on the model. To solve this problem, different smoothing coefficients are set to test the performance of the model on the test set. The minimum value $\sigma_{\min}$ of the smoothing coefficient that expands 2 times each time is set to 10^{-9} , that is, $\sigma_n = 2\sigma_{n-1}$. Then an experiment is carried out, as shown in figure 4.23. Figure 4.23 shows that there is a suitable smoothing coefficient to enhance the predicted performance of the model.

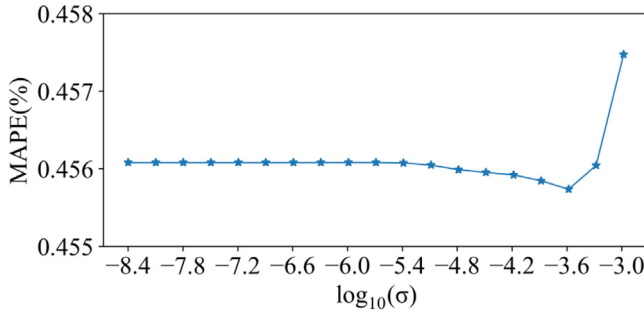


FIG. 4.23 – The MAPE of different smoothing coefficients.

To search for the best smoothing coefficient of the model, the data set is redivided before the next experiment is carried out. The original data set is divided into the training set, validation set, and test set. Then the Artificial Fish Swarm Algorithm (AFSA) is used to optimize the smoothing coefficient on the validation set. To compare the model using AFSA with the previous model, various indicators are obtained to help evaluate the two models. Table 4.5 shows that all indicators

TAB. 4.5 – The evaluation indicators of the GRNN and AFSA-GRNN.

	GRNN	AFSA-GRNN
MAE (mm)	1.400	0.864
RMSE (mm)	1.600	1.500
MAPE (%)	0.1433	0.091

have decreased after the AFSA is used to optimize the original model, which indexes that the optimization algorithm can enhance the performance of the model.

4.6 Robot Arm Time-Series Deep Learning Forecasting Method

Deep learning shows higher forecasting accuracy in practice and can obtain more detailed information on time series. It has been widely used recently. In this section, three deep learning forecasting methods are employed to forecast robot arm time series, including Autoencoder (AE) deep neural network and Deep Belief Network (DBN).

4.6.1 Autoencoder Deep Neural Network Forecasting Algorithm

AE is an unsupervised machine learning algorithm that uses neural networks to obtain important features that can represent high-dimensional input data. Principal Component Analysis (PCA), a traditional linear dimensionality reduction method, captures the direction of high-dimensional data by selecting those axes with the largest variance. However, the linear characteristic of the PCA makes the feature dimension of PCA limited. AE overcomes these limitations with its nonlinear neural network structure and can learn the essential features of data from a wide range of unlabeled raw data.

(1) Algorithm principle

AE is mainly composed of the encoder and the decoder. The encoding process uses an encoding function $f_{\theta}(\mathbf{x}) = s(\mathbf{W}\mathbf{x} + \mathbf{b})$ to encode the input vector $\mathbf{x} \in [0, 1]^{d_x}$ into a new compressed representation $y \in [0, 1]^{d_h}$. s is a nonlinear activation function. The main parameters can be expressed as $\theta = \{\mathbf{W}, \mathbf{b}\}$, the weight $\mathbf{W}$ is a $d_h \times d_x$ dimensional matrix, and the offset $\mathbf{b}$ is a d_h dimensional vector. The decoding function $g_{\theta'}(y) = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$ decodes the compressed representation y into the output $z \in [0, 1]^{d_z}$. If $\mathbf{W}' = \mathbf{W}^T$, the parameter θ' can be expressed as $\theta' = \{\mathbf{W}', \mathbf{b}'\}$ (Baldi, 2012).

Suppose $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \dots, \mathbf{x}^{(n)}\}$ is a group of unlabeled data. First, calculate the middle layer representation $y^{(i)}$ corresponding to each input vector $\mathbf{x}^{(i)}$, and get the reconstructed representation $\mathbf{z}^{(i)}$ in the output layer. The overall cost function of $\mathbf{X}$ and $\mathbf{Z}$ is (Baldi, 2012):

$$J_{\text{AE}}(\theta, \theta') = \frac{1}{n} \sum_{i=1}^n L(\mathbf{x}^{(i)}, g(f(\mathbf{x}^{(i)}))) \quad (4.27)$$

where L is the reconstruction error function. When the sigmoid function is used as the activation function, the cross-entropy cost function is (Baldi, 2012):

$$L(\mathbf{x}, \mathbf{z}) = - \sum_{k=1}^{d_x} [x_k \log z_k + (1 - x_k) \log(1 - z_k)] \quad (4.28)$$

(2) Modeling flow

The modeling flow of the AE deep neural network contains three components, including data processing, training, and forecasting.

- (a) In the data preprocessing component, the robot arm time series is divided into the X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle series.
- (b) In the training component, the parameters of the AE deep neural network are trained.
- (c) In the forecasting component, the trained AE deep neural network is applied to generate forecasting values.

(3) Data processing

Considering the obvious periodicity of the time series data of the robotic arm, the modeling data contain 7907 data, the 1500th–4200th is training data, and the 4201th–5000th is testing data. The training data is used for training the AE deep neural network model, and the testing data is used for forecasting performance evaluation. Set the first five position series to forecast the position of the next moment. The original X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle series are shown in figure 4.24.

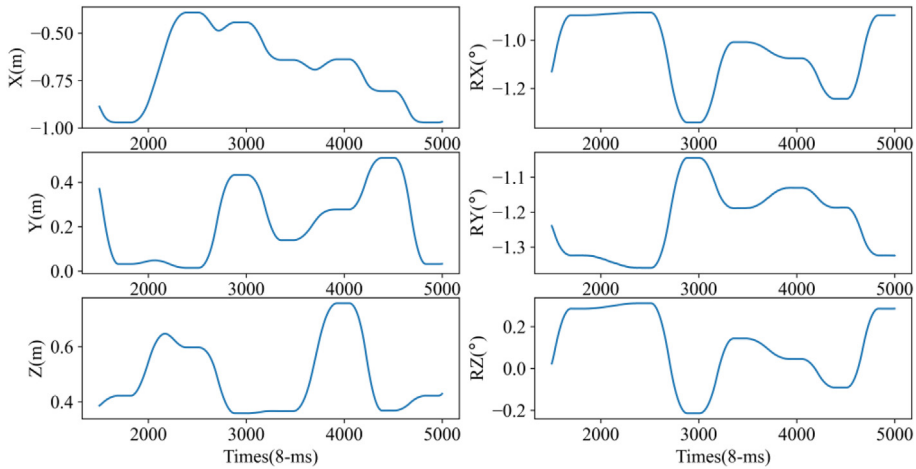


FIG. 4.24 – The original robot arm position time series.

(4) Training and forecasting

To learn the influence of AE on the forecasting improvement, the robot arm time series set is applied to the Long Short-Term Memory (LSTM) network and AE-LSTM respectively, and the prediction results are compared.

Taking the X-position data as an example, the experimental results of the predicted values of the LSTM are shown in figure 4.25.

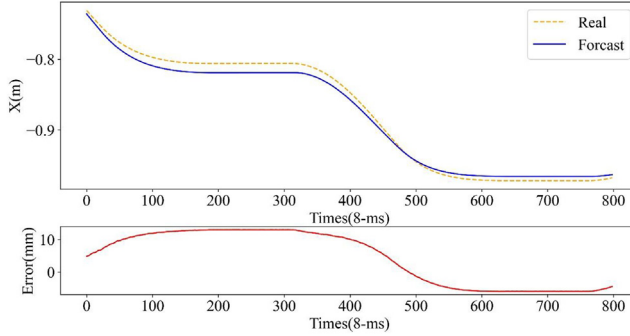


FIG. 4.25 – The true value and predicted value of the LSTM.

The experimental results of the predicted values of the AE-LSTM are shown in figure 4.26.

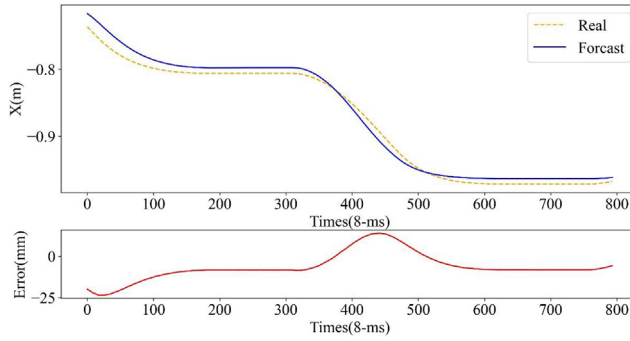


FIG. 4.26 – The true value and predicted value of the AE-LSTM.

It can be found that the prediction accuracy of the LSTM model without AE decomposition is better. To compare their forecasting differences more clearly, the evaluation indexes MAE, RMSE, and MAPE corresponding to the LSTM without AE and the DBN with AE are respectively calculated, as shown in table 4.6. It can be known from table 4.6 that the model with AE may not improve the accuracy of the model. This may be because the selected data is too smooth.

TAB. 4.6 – The evaluation indicators of the LSTM and AE-LSTM.

	LSTM	AE-LSTM
MAE (m)	0.017	0.043
RMSE (m)	0.021	0.046
MAPE (%)	1.836	4.827

4.6.2 Deep Belief Network Forecasting Algorithm

DBN can be used for unsupervised learning and supervised learning. It is a probabilistic generative model.

(1) Algorithm principle

DBN is a combination of a generative model and a discriminative model. Its structure is divided into two parts. The first part is the unsupervised deep Boltzmann machine, which represents the generative model and completes the feature extraction of the data (Hua *et al.*, 2015). The deep Boltzmann machine is obtained by stacking any number of layers of Restricted Boltzmann Machines (RBM). It is trained by unsupervised layer-by-layer training. The second part is the supervised back propagation neural network model, which represents the discriminant model. The DBN structure formed by stacking two layers of RBM is shown in figure 4.27 (Hua *et al.*, 2015).

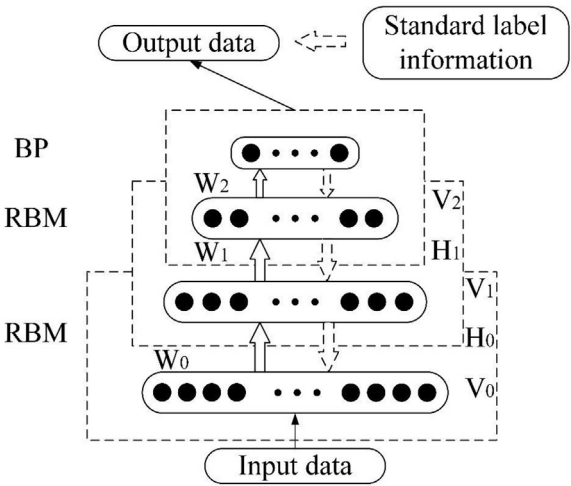


FIG. 4.27 – Structure of the DBN.

RBM is obtained by specialization of the Boltzmann machine, which belongs to the undirected graph model. Its structure is divided into two layers, the visible layer and the hidden layer, which are used for the input of feature data and feature reconstruction respectively (Hinton, 2012).

RBM is a probability model based on energy, and the process of training RBM is the process of minimizing the energy function. Assume that the state vector of the RBM visible layer is $\mathbf{v} = [v_1, v_2, \dots, v_i, \dots, v_n]$, and the state vector of the hidden layers are $\mathbf{h} = [h_1, h_2, \dots, h_j, \dots, h_m]$ and $\theta = \{w_{ij}, a_i, b_j\}$. $w_{ij} \in W_{n \times m}$ is the weight in the connection matrix of the visible layer and the hidden layer. a_i is the node threshold of the visible layer. b_j is the hidden layer node threshold. Given the training samples of the system, the energy function of the RBM model is defined as (Fischer and Igel, 2012):

$$E(\mathbf{v}, \mathbf{h} | \theta) = - \sum_{i=1}^n a_i v_i - \sum_{j=1}^m b_j h_j - \sum_{i=1}^n \sum_{j=1}^m h_j w_{ij} v_i \quad (4.29)$$

According to the energy function, the joint probability distribution between the visible layer and the hidden layer can be obtained (Fischer and Igel, 2012):

$$P(\mathbf{v}, \mathbf{h} | \theta) = \frac{1}{Z(\theta)} e^{-E(\mathbf{v}, \mathbf{h} | \theta)} \quad (4.30)$$

where $Z(\theta) = \sum_{\mathbf{v}, \mathbf{h}} e^{-E(\mathbf{v}, \mathbf{h} | \theta)}$ is the distribution function.

Considering the characteristics of the RBM structure, when the visible layer node status is determined, the activation probability conditions of each node in the hidden layer are independent. The conditional probabilities of the visible layer v and the hidden layer h are respectively expressed as (Fischer and Igel, 2012):

$$P(\mathbf{v} | \mathbf{h}) = \prod_{i=1}^n P(v_i | \mathbf{h}) \quad (4.31)$$

$$P(\mathbf{h} | \mathbf{v}) = \prod_{j=1}^m P(h_j | \mathbf{v}) \quad (4.32)$$

where $P(v_i | \mathbf{h})$ is the probability that a single visible layer unit is activated when the state of all hidden layer nodes $\mathbf{h}$ is determined. The definition of $P(v_i | \mathbf{h})$ is as follows (Fischer and Igel, 2012):

$$p(v_i = 1 | \mathbf{h}, \theta) = \text{sigmoid} \left(a_i + \sum_{j=1}^m w_{ij} h_j \right) \quad (4.33)$$

$$p(h_j = 1 | \mathbf{v}, \theta) = \text{sigmoid} \left(b_j + \sum_{i=1}^n w_{ij} v_i \right) \quad (4.34)$$

(2) Modeling flow

The modeling flow of the DBN belief network contains three components, including data processing, training, and forecasting.

- (a) Data preprocessing. The robot arm time series is divided into the X-position, Y-position, Z-position, RX-angle, RY-angle, and RZ-angle series.

- (b) Training. The parameters of the DBN belief network are trained.
- (c) Forecasting. The trained DBN belief network is applied to generate forecasting values.

(3) *Data processing*

By analyzing the time series data set of the robot arm, it can be found that the series shows obvious periodicity. Taking the X-position series as an example, a total of 3500 data are selected as training data and testing data. The former 2700 data are used to train the deep predictors, and the last 800 data are used to test the forecasting performance. Set the first five position series to forecast the position of the next moment. The division of data can be seen in figure 4.28.

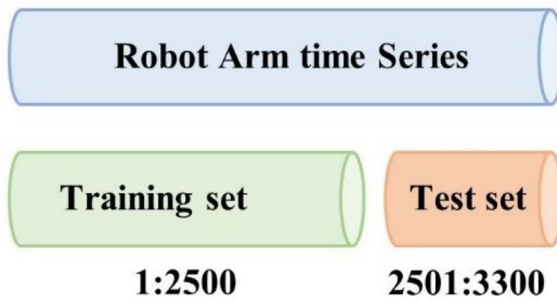


FIG. 4.28 – The split of training set and test set for DBN.

(4) *Training and forecasting*

The selected training data are used to construct the DBN belief network. Set the number of hidden layer nodes to 5 and the number of iterations to 300. After the training, the test set data is input into the model, and the true value is compared with the predicted value. Taking the X-position data as an example, the experimental results of the predicted values and the true values are shown in figure 4.29. It can be known from figure 4.29 that there is not much difference between the predicted value and the true value.

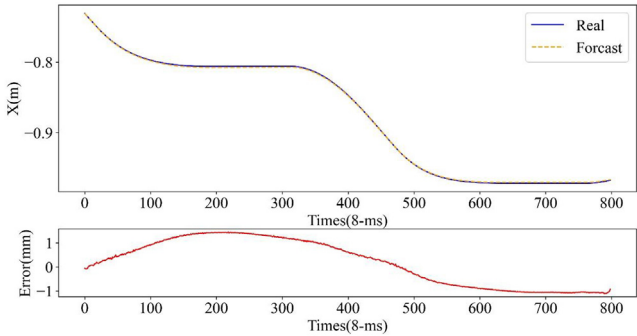


FIG. 4.29 – The true value and predicted value of DBN.

For further improving the forecasting accuracy, the Variational Mode Decomposition (VMD) method is introduced. The experimental results of the DBN model with the VMD are shown in figure 4.30.

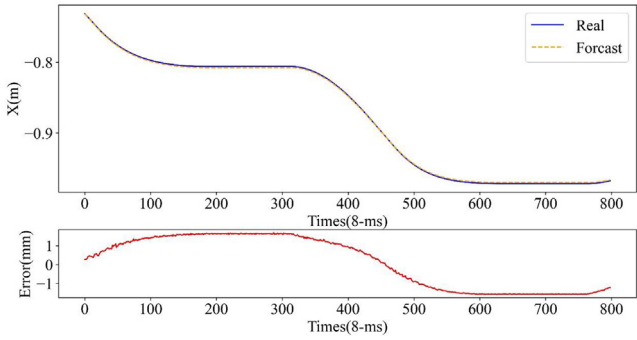


FIG. 4.30 – The true value and predicted value of the DBN with the VMD.

It can be found that the prediction accuracy of the DBN model with the VMD decomposition is better. To compare their forecasting differences more clearly, the evaluation indexes MAE, RMSE, and MAPE corresponding to the DBN without VMD and the DBN with the VMD are respectively calculated, as shown in table 4.7. From table 4.7, it can be known that all evaluation indicators have decreased after adding the decomposition. It shows that the decomposition algorithm can enhance the performance of model forecasting.

TAB. 4.7 – The evaluation indicators of the DBN and VMD-DBN.

	DBN	VMD-DBN
MAE (mm)	1.305	1.201
RMSE (mm)	1.362	1.304
MAPE (%)	0.152	0.136

4.7 Comparative Analysis of Forecasting Performance

The above experiments are carried out on the forecasting effect of robot arm time series from the statistical method, intelligent method, and deep learning method. Three methods are compared below.

4.7.1 Analysis of Statistical Methods

According to the experimental results in section 4.4, the following conclusions can be drawn.

- (a) The two models adopted in the experiment are based on the ARIMA model. Although the ARIMA model has good performance on the data set, the process of establishing the ARIMA model is complicated, and the Pandit–Wu prediction algorithm can simplify this process.
- (b) It can be known from table 4.2 that the prediction performance of the ARIMA model decreased significantly after adding an artificial noise signal. After the data are processed by the Kalman filter, the fluctuation of the data is significantly reduced, and the indicators are significantly improved, indicating that the Kalman filter can significantly enhance the anti-noise performance of the model.

4.7.2 Analysis of Intelligent Methods

According to the experimental results in section 4.5, the following conclusions can be drawn.

- (a) Three intelligent forecasting algorithms can accurately predict the robot arm time series.
- (b) The parameters of the three intelligent prediction algorithms have a great influence on the forecasting accuracy of the model. For different data, different parameters need to be set to make the prediction accuracy of the model reach a higher level.
- (c) According to the experimental results of sections 4.5.1 and 4.5.3, the forecasting accuracy of the model with PSO and AFSA is better than that of the original model as shown in table 4.8. This shows that using the optimization algorithm to optimize the parameters in the intelligent forecasting algorithm can significantly enhance the forecasting accuracy of the intelligent algorithm.

TAB. 4.8 – Comparison of prediction performance before and after optimization.

	Before optimization	After optimization	Improving percentages
RELM			
MAE (mm)	5.141	3.204	37.67%
RMSE (mm)	5.883	4.764	19.02%
MAPE (%)	0.079	0.051	35.44%
GRNN			
MAE (mm)	1.400	0.864	38.28%
RMSE (mm)	1.600	1.500	6.25%
MAPE (%)	0.143	0.091	36.36%

4.7.3 Analysis of Deep Learning Methods

According to the experimental results in section 4.6, the following conclusions can be drawn.

The superposition of deep learning methods may not necessarily improve the forecasting ability of the model. According to the forecasting results of section 4.6.1,

it can be found that the forecasting results of the LSTM model with AE are worse than those using LSTM alone. This may be because the selected data set is relatively smooth, so there is no need to choose multiple hybrid deep learning methods.

Decomposition can improve the forecasting ability of the model to a certain extent. According to the forecasting results of section 4.6.2, it can be found that the forecasting effect of the DBN model with VMD is better than using DBN alone. This is because decomposition can mine more information about the original time series.

4.8 Robot Arm Positioning Control Based on Arm Forecasting

The forecasting of robot arm time series is significant to robot control. The industry gradually tends to automate development, and industrial robots account for an increasing proportion of the market. Industrial robots can replace workers to do some repetitive single long-term industrial production operations, or operate in dangerous environments full of chemicals, radiation, etc., to maximize the safety of workers. The application of industrial robots greatly saves human resources, increases the efficiency and quality of work and production, and reduces the loss in the production process. The applicability of industrial robots has been continuously developed, and the tasks that can be completed in various fields and environments are becoming increasingly complex and diverse. It has not only greatly promoted the manufacturing industry, but also played an important role in the military, medical, and automobile manufacturing fields.

The mechanical arm of an industrial robot has partial operations similar to a human arm and wrist. Through the programming and control of a computer, the functions and operations of the human arm are basically realized. At present, the main components of the mechanical arm of an industrial robot are the actuator, the wrist, the arm, the stand column, the walking mechanism, and the machine base. The systems of the robot arm are shown in figure 4.31.

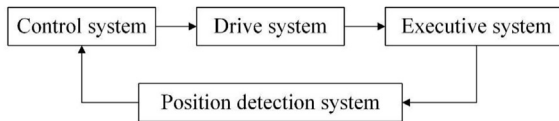


FIG. 4.31 – The main components of the robot arm.

The control system controls the operation mode of the arm to ensure the safe and efficient operation of the industrial robot arm. The drive system should minimize the drive response time of the robot arm so that the overall operation of the drive part of the robot arm is streamlined and safe. The executive system can make the robot arm work under the control signal, and the executive ability is very strong. The position detection system makes the motion of the robot arm more accurate to achieve the desired effect. Realizing the prediction of the operation of the robot arm is very

important in the control of the industrial robot, which can minimize the damage rate of the industrial robot arm and is easy to maintain so that it can operate accurately.

Besides, robots also play an important role in the laboratory. Using a robot arm, the robot can replace manual laboratory equipment operations in a fixed workspace, such as the preparation of pharmaceutical reagents, the injection of liquid medicine into test tubes, and the test tubes into special containers. In this process, the positioning accuracy of the robot arm is very important. If the coordinate information of the robot arm is not accurate, problems such as experimental operation errors and reagent spills may occur. The prediction of robot arm position coordinates can alleviate these negative positioning phenomena. The predicted position data can correct the positioning error and improve the positioning accuracy. When the positioning system fails, the predicted position can be used as the current position to realize the predictive control of the robot arm.

4.9 Conclusions

This chapter explains several robot arm times forecasting models. Three different methods are introduced, including statistical methods, intelligent methods, and deep learning methods. The experimental conclusions are as follows:

- (a) Based on the performance effects of all the above models, the effects of the statistical forecasting model and the intelligent forecasting model are better than the deep learning model. This may be because when the number of samples in the data set is relatively small, or the form of change of the sample data is relatively simple, the deep learning model lacks data and has a complex structure.
- (b) Both data decomposition and optimization algorithms can improve the forecasting accuracy of the model. Data decomposition can make the form of data fluctuations simpler and reduce the difficulty of model forecasting. The optimization algorithm can optimize the parameters of the model, find the optimal parameters that are suitable for the model, and significantly enhance the prediction accuracy of the model.

Finally, the application of the robot arm time series prediction algorithm in industrial robots and laboratory robots is analyzed. The prediction of the robot arm time series can achieve more accurate robot arm control operation and timely diagnosis according to the prediction results.

References

- Baldi P. (2012) *Autoencoders, unsupervised learning, and deep architectures*. Paper presented at the Proceedings of ICML workshop on unsupervised and transfer learning.
- Burnham K. P., Anderson D. R. (2004) Multimodel inference: Understanding AIC and BIC in model selection, *Sociol. Methods Res.* **33**(2), 261–304.
- Cantelli L., Muscato G., Nunnari M., Spina D. (2015) A joint-angle estimation method for industrial manipulators using inertial sensors, *IEEE/ASME Trans. Mechatron.* **20**(5), 2486–2495.

- Chen T., He T., Benesty M., Khotilovich V., Tang Y., Cho H. (2015) Xgboost: Extreme gradient boosting, *R package version 0.4-2*, **1**(4), 1–4.
- Cheng P., Oelmann B. (2009) Joint-angle measurement using accelerometers and gyroscopes—A survey, *IEEE Trans. Instrum. Meas.* **59**(2), 404–414.
- Fischer A., Igel C. (2012) *An introduction to restricted Boltzmann machines*. Paper presented at the Iberoamerican congress on pattern recognition.
- Georgopoulou A., Michel S., Vanderborght B., Clemens F. (2021) Piezoresistive sensor fiber composites based on silicone elastomers for the monitoring of the position of a robot arm, *Sens. Actuators A: Phys.* **318**, 112433.
- Hinton G. E. (2012) A practical guide to training restricted Boltzmann machines, *Neural networks: Tricks of the trade*. Springer, pp. 599–619.
- Hu N.-Z., Su C.-H. S., Gong C.-S. A., Lee C.-J., Chen Y.-S., Yang C.-H., ... Chuang J.-T. (2019) *Machine learning approach for robot diagnostic system*. Paper presented at the 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE).
- Hua Y., Guo J., Zhao H. (2015) *Deep belief networks and deep learning*. Paper presented at the Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things.
- Huang G.-B., Zhou H., Ding X., Zhang R. (2011) Extreme learning machine for regression and multiclass classification, *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **42**(2), 513–529.
- Huang P. W., Chung K.-J. (2019) *The prediction of positioning shift for a robot arm using machine learning techniques*. Paper presented at the 2019 14th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT).
- Huang Y.-C., Yang Z.-S., Liao H.-S. (2019) Labeling confidence values for wafer-handling robot arm performance using a feature-based general regression neural network and genetic algorithm, *Appl. Sci.* **9**(20), 4241.
- Jassemi-Zargani R., Necsulescu D. (2002) Extended Kalman filter-based sensor fusion for operational space control of a robot arm, *IEEE Trans. Instrum. Meas.* **51**(6), 1279–1282.
- Jhang L.-H., Santiago C., Chiu C.-S. (2017) *Multi-sensor based glove control of an industrial mobile robot arm*. Paper presented at the 2017 International Automatic Control Conference (CACS).
- Kalman R. E. (1960) A new approach to linear filtering and prediction problems.
- Lee Y.-H., Kim K.-J., Lee S.-I., Kim D.-J. (2019) Seq2Seq model-based prognostics and health management of robot arm, *J. Korea Inst. Inf. Electron. Commun. Technol.* **12**(3), 242–250.
- Pandit S. M., Wu S. M., Šmits T. I. (1984) *Time series and system analysis with applications* (S. M. Pandit, S.-M. Wu, Eds). Acoustical Society of America.
- Pradeep V., Konolige K., Berger E. (2014) *Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach*. Paper presented at the Experimental robotics.
- Qiao G. (2019) *Advanced sensor and target development to support robot accuracy degradation assessment*. Paper presented at the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE).
- Qiao G., Weiss B. A. (2017) *Accuracy degradation analysis for industrial robot systems*. Paper presented at the International Manufacturing Science and Engineering Conference.
- Riazi M., Zaiane O., Takeuchi T., Maltais A., Günther J., Lipsett M. (2019) *Detecting the onset of machine failure using anomaly detection methods*. Paper presented at the International Conference on Big Data Analytics and Knowledge Discovery.
- Sareh S., Noh Y., Li M., Ranzani T., Liu H., Althoefer K. (2015) Macrobend optical sensing for pose measurement in soft robot arms, *Smart Mater. Struct.* **24**(12), 125024.
- Specht D. F. (1991) A general regression neural network, *IEEE Trans. Neural Networks* **2**(6), 568–576.
- Welch G., Bishop G. (1995) An introduction to the Kalman filter.
- Widmaier F., Kappler D., Schaal S., Bohg J. (2016) *Robot arm pose estimation by pixel-wise regression of joint angles*. Paper presented at the 2016 IEEE International Conference on Robotics and Automation (ICRA).
- Zhang L., Zhang D. (2016) Evolutionary cost-sensitive extreme learning machine, *IEEE Trans. Neural Networks Learn. Syst.* **28**(12), 3045–3060.

Chapter 5

Unmanned Vehicle Time Series Predictive Control

5.1 Introduction

As an important part of the intelligent transportation system, unmanned vehicles can effectively reduce traffic accidents and traffic congestion. An unmanned vehicle is in a complex traffic scene, and the behavior of surrounding vehicles and pedestrians is full of uncertainty, this requires the unmanned vehicle to be able to predict the future evolution of the traffic scene to perform correct warnings and decisions (Baek *et al.*, 2020; Schreier *et al.*, 2014). One of the main problems of unmanned vehicles operating in a complex traffic scene is to interact safely and reasonably with other participants. Vehicle trajectory forecasting is a prerequisite for unmanned vehicle motion planning and combines the interaction between vehicles in collision detection and trajectory evaluation (Sharma *et al.*, 2019). Accurate vehicle trajectory forecasting can reduce the probability of danger while driving, and can also improve the driving efficiency and comfort of unmanned vehicles (Lefèvre *et al.*, 2014).

The vehicle trajectory is constrained by prior knowledge of road geometry, traffic signals, traffic rules, etc., it also depends on the uncertain posterior knowledge of other traffic participants and subjective driving intentions. Regarding the complexity of the driving behavior model, the vehicle trajectory prediction method consists of three parts, which are trajectory prediction based on physics, trajectory prediction based on maneuver, and trajectory prediction based on interactive perception, as shown in figure 5.1 (Jeong and Yi, 2019).

(1) *Physics-based trajectory forecasting*

The physics-based method represents the vehicle as a dynamic entity governed by the laws of physics. The motion trend of the vehicle is predicted by dynamics and kinematics models, and the control state, vehicle characteristics, and road factors are linked to the state evolution of the vehicle. This method satisfies the assumption that the current state of the vehicle is deterministic and the control input of the kinematics model is constant, but the long-term forecasting results are unreliable due to ignoring the uncertainty and incompleteness of the vehicle state and control input (Bahram *et al.*, 2016).

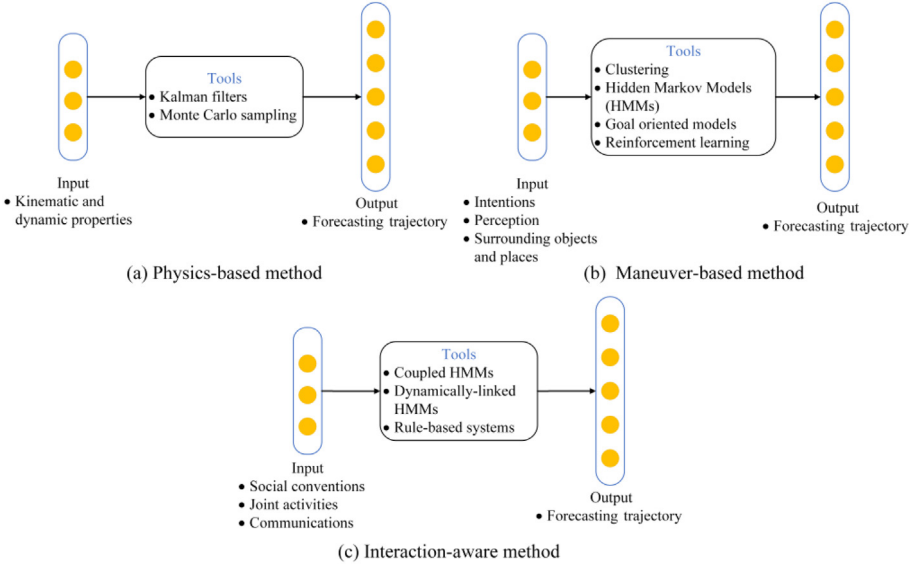


FIG. 5.1 – Classification of trajectory forecasting methods.

Some studies have modeled the uncertainty of the vehicle state. Researchers used Gaussian mixture matrices to model the uncertainty, and a switching Kalman filter was used to represent possible vehicle motion models (Veeraraghavan *et al.*, 2006). To solve the probability forecasting of road traffic sense, Markov chain abstraction and Monte Carlo simulation were used to assess the probabilistic occupancy of other participants and collision risk (Althoff and Mergel, 2011). The physics-based method only focuses on the constraints of vehicle motion characteristics on motion trajectory, and cannot predict the vehicle maneuver due to changes in the driving scene, so it is limited to low-speed short-term forecasting.

(2) Maneuver-based trajectory forecasting

The maneuver-based method assumes that vehicle movement corresponds to a series of predefined maneuvering behaviors, and the vehicle trajectory corresponding to each behavior has similar characteristics. Once the behavior of the vehicle is determined, the future vehicle trajectory can be predicted, thereby dividing the forecasting process into behavior recognition and trajectory forecasting (Ortiz *et al.*, 2011). A lot of information in the dynamic scene is used to estimate vehicle behavior. The priori information includes ego vehicle state and road geometry information, and the posterior information includes vehicle interaction, driver driving style, body posture, etc. Many machine learning methods are widely used in vehicle behavior recognition, such as support vector machines for driving behavior classification, and Markov random field, belief networks for maneuvering probability distribution calculation (Klingelschmitt *et al.*, 2014; Morris *et al.*, 2011).

After the possible driving behavior is obtained, the predicted trajectory can be solved from the kinematic motion model, with the input controls corresponding to

the recognized driving behavior (Kumar *et al.*, 2013). In order to consider the uncertainty of vehicle state and maneuver intention, the Gaussian process or rapidly exploring random tree can be used to explore the potential execution space. The maneuver-based forecasting method decomposes vehicle motion into predefined driving action sequences to identify and classify driving intentions. Compared with the physics-based forecasting method, the forecasting results are more stable and accurate and get longer forecasting.

(3) Interaction-aware trajectory forecasting

The driving of a vehicle is affected by other traffic participants, the interaction-aware method considers the dependence between vehicles to better explain their motion. Features related to multi-vehicle interaction can reflect how the motion of a vehicle is affected by interactive behaviors (for example, anti-collision, interaction between vehicles, traffic rules, social customs, etc.), so the interaction-aware method can better understand future traffic conditions.

Dynamic Bayesian network is a common forecasting method that directly considers interactive features (Gindele *et al.*, 2015; Kasper *et al.*, 2012). Deep learning methods can implicitly consider interactive features. Convolutional Neural Network (CNN) was used to predict vehicle lane-changing intention on the highway, the input of the network is the occupancy map, and the output is the possible lane-changing of the vehicle in front (Lee *et al.*, 2017). In addition, a Gaussian mixture model parameterized by a deep neural network was presented to the motion of a group of vehicles on the highway, and experiments have shown that it is better than the CNN and traditional intelligent driver model (Lenz *et al.*, 2017). Recurrent Neural Network (RNN) can also be applied to vehicle motion forecasting, Long Short-Term Memory (LSTM) is applied to predict the intention of vehicles at the intersection and the position of vehicles in the occupancy grid map (B. Kim *et al.*, 2017; Phillip *et al.*, 2017).

Vehicle trajectory forecasting is a time series forecasting problem actually, several statistical, intelligent, and deep learning forecasting models will be introduced. The research flowchart of unmanned vehicle trajectory series modeling is divided into four parts, including data acquisition and analysis, time series modeling, application, and conclusion. The whole framework of models used in this chapter is shown in figure 5.2.

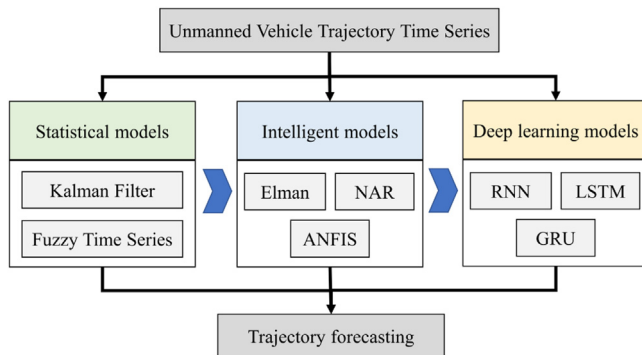


FIG. 5.2 – Framework of models used in chapter 5.

5.2 Unmanned Vehicle Time Series Measurement

Environment perception and positioning are the basis of unmanned vehicles, sensors are used to realize the perception of the surrounding environment and motion state. Vehicle positioning and navigation are completed by analyzing and fusing the time series data of multi-sensors to realize motion planning and decision-making. The multi-sensor fusion technology of unmanned vehicles realizes the coverage of multiple targets and reflects environmental information more accurately (Kocić *et al.*, 2018). The type of sensor determines the quantity and quality of information acquired by the vehicle, as shown in figure 5.3, there are mainly three kinds of sensors for high-precision positioning of vehicles, including vision sensor, lidar, and inertial navigation system.

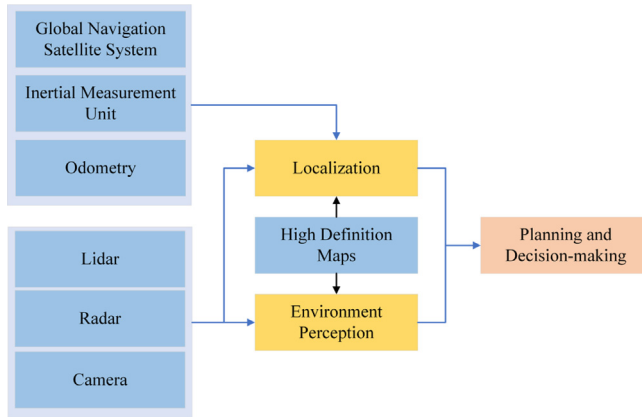


FIG. 5.3 – Classification of multi-sensor perception in unmanned vehicle.

Global Navigation Satellite System (GNSS) and Inertial Measurement Unit (IMU) provide global position and inertial information for the unmanned vehicle, and odometry is installed on the wheel to obtain the driving speed, to update the position of the vehicle in the map in real-time (Morales *et al.*, 2016). Vision sensors can acquire optical images and record the contours, textures, and color distributions of objects, vision positioning is to match the sensor data with three-dimensional map data. Therefore, the camera can be used to complete target recognition and tracking, including lane detection, pedestrian and vehicle recognition, and local path planning (Zhang *et al.*, 2020). Lidar can directly obtain the distance information from the object, and illumination change does not affect the ranging accuracy, it is used to complete positioning, obstacle detection, and environmental reconstruction (Peron *et al.*, 2019). Lidar positioning is to map the sensor data with the map based on point cloud registration.

All these measurement methods can produce vehicle trajectory series. The introduced models in this chapter are data-driven. The historical trajectory is the only need for data-driven models, and the trajectory data includes the longitude and latitude of the current location of the vehicle.

5.3 Unmanned Vehicle Time Series Uncertainty Analysis

The uncertainty of the vehicle trajectory time series is related to two aspects, including environmental uncertainty and measurement uncertainty.

(1) *Environmental uncertainty*

The unmanned vehicle is in an unstructured environment, and the environment perception information is complex and changing. There are obvious characteristics among different environmental types in an unstructured environment. Due to the inconsistency of terrain, the irregularity of obstacles, and the openness of the driving scene, there are problems and challenges in recognition and sensing. These problems include the impact of different weather conditions on the performance of sensors, as well as the movement of objects, illumination change, and environmental noise. For example, the GNSS cannot provide accurate positioning due to multipath and non-line-of-sight propagation effects in a complex environment, vision sensors have a great error when illumination changes drastically.

These complex information changes will interfere with the sensing system, leading to uncertainty in sensor measurement data. Multi-sensor fusion technology can reduce this uncertainty through the redundancy and complementary of information, which is very useful for uncertain information processing.

(2) *Measurement uncertainty*

The measurement uncertainty comes from the cumulative error of sensors and the error of data acquisition. In the vehicle sensing system, due to the influence of factors such as noise and bias of the accelerometer and gyroscope, data errors will accumulate over time. Data acquisition will be interfered with by different factors, making the collected data incomplete or inaccurate. The main error of the GNSS data is the satellite ephemeris error, which represents the difference between the actual position and spatial position. The second is signal propagation error, which is caused by the multipath effect, tropospheric refraction, ionosphere refraction, etc.

5.4 Unmanned Vehicle Time Series Statistical Forecasting Method

In the analysis of sequence trends, statistical models are often used. In this section, two different statistical methods are introduced for vehicle trajectory series forecasting, including Kalman filter forecasting and fuzzy time series forecasting.

5.4.1 *Kalman Filter Forecasting Algorithm*

(1) *Theoretical basis*

Kalman filter is used to estimate the state of system by using a state equation and observation equation with observation data. This characteristic has been widely

studied in various vehicle position or trajectory forecasting, such as forecasting of ball position in robot soccer, localization of mobile robots, and vehicle position forecasting (Abbas *et al.*, 2020; Greenberg and Tan, 2020; Mo *et al.*, 2016; Cheng and Li, 2015).

The process of the Kalman filter includes forecasting and correction of the state vector. The forecasting process is based on the essence of the model to forecast the system state with an estimated value at time $t - 1$ and input control. Then, the transformation between forecasting value, measured value, gain matrix, and noise is established, to realize the correction of forecasting value. The modeling flows of the Kalman filter are described as follows:

- (a) Build system model.
- (b) Set initialization parameters.
- (c) Forecast the state at time t based on the state at time $t - 1$.
- (d) Estimate the system forecasting error at time t according to the system forecasting error at time $t - 1$.
- (e) Calculate the Kalman gain matrix.
- (f) Calculate the optimal estimation of the system.
- (g) Calculate system forecasting error at the current moment
- (h) Back to step (c) and start the forecasting at time $t + 1$.

The state equation and observation equation of system in the Kalman filter model can be expressed by the following equations (Śmieszek and Dobrzańska, 2015):

$$\begin{cases} x_t = Ax_{t-1} + Bu_{t-1} + w_{t-1} \\ z_t = Hx_t + v_t \end{cases} \quad (5.1)$$

where x_t , x_{t-1} is a vector of the system state, A and B represent state transition matrix and control matrix respectively, u_{t-1} is the input of the system, w_{t-1} is system noise, z_t is measured value, H is conversion matrix from state vector to the measured value, v_t is measurement noise.

The time update equation and state update equation of the Kalman filter can be expressed as the following equations (Śmieszek and Dobrzańska, 2015):

$$\begin{cases} \hat{x}_t = A\hat{x}_{t-1} + Bu_{t-1} \\ P_t = AP_{t-1}A^T + Q \end{cases} \quad (5.2)$$

$$\begin{cases} K_t = (P_t H^T) / (HP_t H^T + R) \\ \hat{x}_t = \hat{x}_t + K_t(z_t - H\hat{x}_t) \\ P_t = (I - K_t H)P_t \end{cases} \quad (5.3)$$

where $\hat{x}_t$ is priori state estimation, P_t is priori estimated covariance, P_t , P_{t-1} are posterior estimated covariance, Q is system noise covariance, K_t is the gain matrix, R is measurement noise covariance, $\hat{x}_t$, $\hat{x}_{t-1}$ are posterior state estimation.

The gain matrix K_t can be obtained by the calculation of the state update equation, then the measured value z_t can be obtained. After the calculation of the

time update and state update at the current time is completed, the posterior estimation can be obtained and the next iteration is started. The prior estimation of the next iteration is the posterior estimation of the previous. The overall calculation process is a recursive process to obtain the forecasting trajectory.

(2) Modeling steps

A. Data preprocessing

The vehicle trajectory describes the format of taxi Global Positioning System reports on Feb 20, 2007, in Shanghai, China (Liu *et al.*, 2010). In this section, the vehicle trajectory of a taxi from 08:00:00 to 18:00:00 is used for time series modeling, the published data is sampled non-uniformly, so these data are resampled with a sampling time of 10 s, and a total of 3600 data are generated in these 10 h. The original trajectory contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series are shown in figure 5.4.

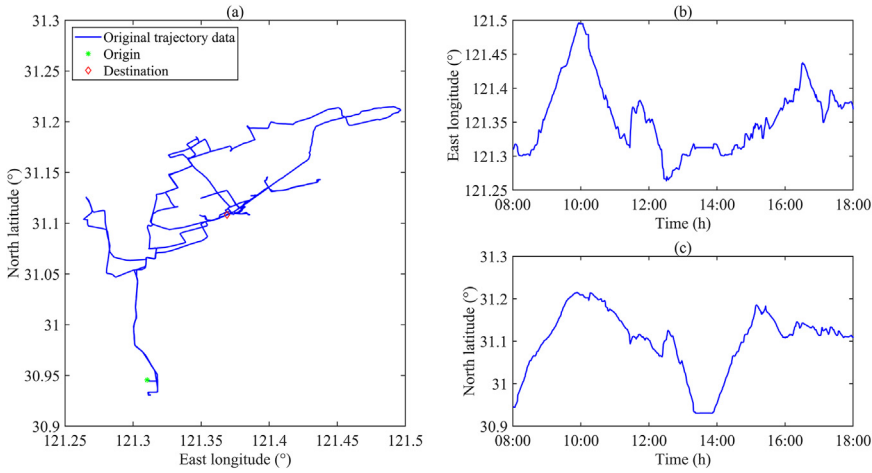


FIG. 5.4 – The original trajectory time series for Kalman Filter experiments: (a) trajectory series, (b) east longitude series, (c) north latitude series.

B. Forecasting

The Kalman filter forecasting is essentially a recursion of single-step forward prediction. The current state and control of the system are used to forecast the next state, and then the system is corrected and updated for the next forecasting, which is an iterative process.

(3) Forecasting results

The performance of the Kalman filter forecasting model is presented in figure 5.5.

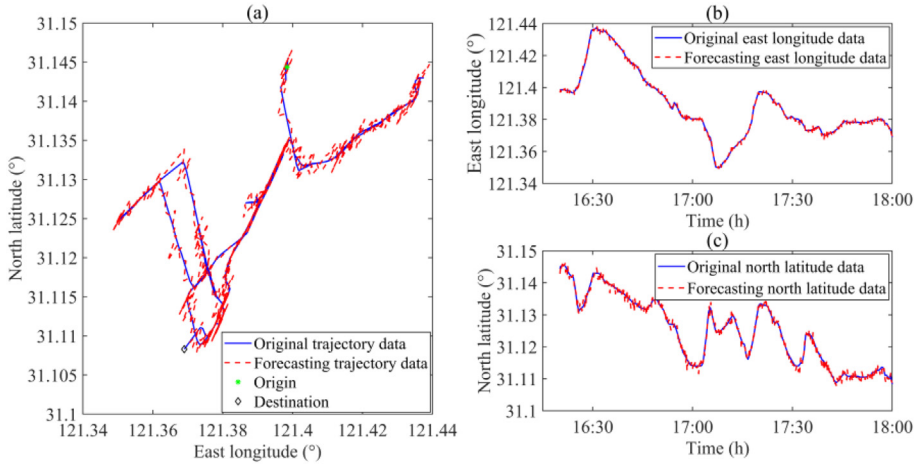


FIG. 5.5 – The forecasting results of the Kalman filter model: (a) trajectory series, (b) east longitude series, (c) north latitude series.

The Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) are usually used to evaluate the performance of the model.

These evaluation indices the MAE, MAPE, and RMSE will also be used in the following sections. The evaluation indices are shown in table 5.1.

TAB. 5.1 – The evaluation indices of the Kalman filter forecasting model.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0008	0.0006	0.0010
North latitude series	0.0008	0.0024	0.0010

5.4.2 Fuzzy Time Series Forecasting Algorithm

(1) Theoretical basis

The fuzzy time series model combines fuzzy set theory and time series model, creating a new research perspective for incomplete and inaccurate data. With this advantage, it has been used in various engineering fields, such as trajectory estimation of moving objects, and damage forecasting (Don *et al.*, 2019; Farzaneh *et al.*, 2014). The principles of modeling include a) define and divide domain, b) define fuzzy set and fuzz historical data, establish the fuzzy logic relationship, c) determine fuzzy rules, and d) defuzzification and output forecasting results.

The domain U is divided into n intervals, that is $U = \{u_1, u_2, \dots, u_n\}$. There is a set A , which belongs to the fuzzy set on the domain U (Bose and Mali, 2019):

$$A = \frac{f_A(u_1)}{u_1} + \frac{f_A(u_2)}{u_2} + \dots + \frac{f_A(u_n)}{u_n} \quad (5.4)$$

where f_A is the fuzzy membership function.

The model is generated on the basis of the following definitions (Bose and Mali, 2019):

Definition 1: R is a set of real numbers, $Y(t)$ ($t = \dots, 0, 1, 2, \dots$) is a subset of R , $f_i(t)$ ($i = 1, 2, \dots$) is a fuzzy set defined on the domain of $Y(t)$. The series $F(t)$ is a set consisting of $f_i(t)$ ($i = 1, 2, \dots$).

Definition 2: $R(t-1, t)$ is a function that satisfies $F(t) = F(t-1) \times R(t-1, t)$, which means $F(t)$ is a first-order model, and $F(t)$ is only caused by $F(t-1)$.

Definition 3: $F(t)$ is a fuzzy time series on the domain U , if $F(t)$ is affected by $F(t-1), F(t-2), \dots, F(t-n)$, that means $F(t)$ is a n -order fuzzy time series.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series are shown in figure 5.4 in section 5.4.1. Then a mapping from 4 sample data points is created to forecast value $x(t)$ by training data sets $[x(t-4), x(t-3), x(t-2), x(t-1)]$. This means four successive known time-series values are used to forecast the next value.

B. Building fuzzy inference system

A type-2 system is constructed with four inputs, each input has three triangular membership functions. Initially, eliminate the footprint of uncertainty for each input membership function by setting each lower membership function equal to its corresponding upper membership function. The structure of the initial model is shown in figure 5.6.

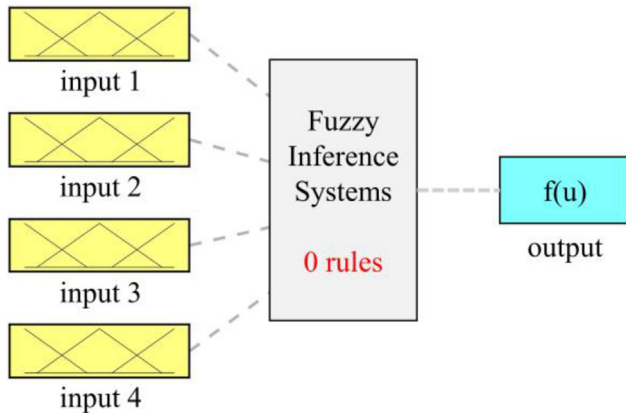


FIG. 5.6 – The structure of the initial type-2 fuzzy inference system.

C. Tuning fuzzy inference system

Tuning fuzzy inference system with train data with the following three steps:

(a) Learning rules

Learning rules with particle swarm optimization to trained fuzzy inference system.

(b) Tuning upper membership function parameters

Tuning upper membership function parameters and retrained fuzzy inference system.

(c) Tuning lower membership function parameters

Tuning lower membership function parameters and retrained model. After tuning the model, the structure of the trained fuzzy inference system is shown in figure 5.7.

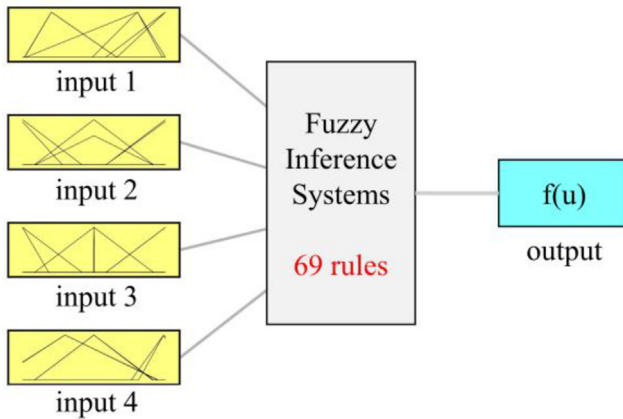


FIG. 5.7 – The structure of the trained type-2 fuzzy inference system.

(3) Forecasting results

The performance of the trained fuzzy inference system is presented in figure 5.8.

The evaluation indices of corresponding forecasting results are shown in table 5.2.

5.5 Unmanned Vehicle Time Series Intelligent Forecasting Method

Intelligent methods can predict more accurately when compared to traditional statistical methods. In this section, three widely used intelligent methods are selected to forecast vehicle trajectory time series, including Elman neural network,

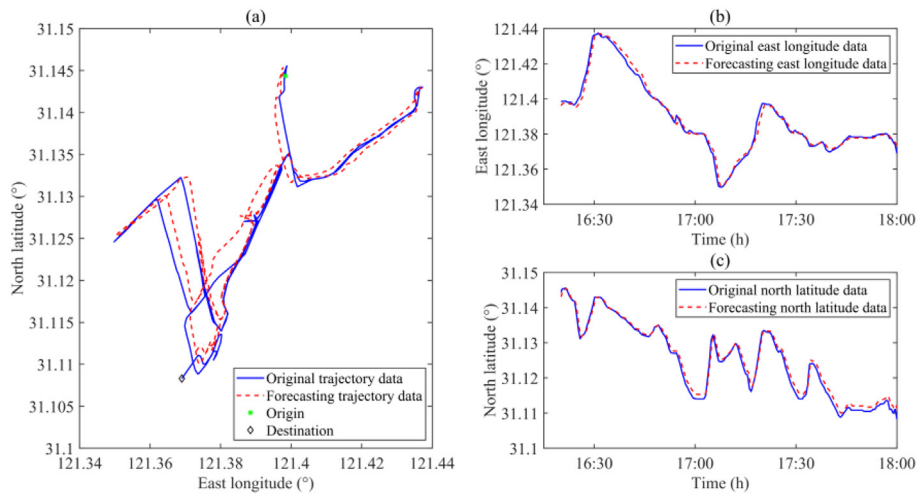


FIG. 5.8 – The forecasting results of the fuzzy inference system: (a) trajectory series, (b) east longitude series, (c) north latitude series.

TAB. 5.2 – The evaluation indices of fuzzy time series model.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0015	0.0012	0.0021
North latitude series	0.0010	0.0032	0.0012

Nonlinear Autoregressive (NAR), and Adaptive Network-based Fuzzy Inference System (ANFIS).

5.5.1 Elman Neural Network Forecasting Algorithm

(1) Theoretical basis

Elman neural network is a network that can realize dynamic nonlinear mapping between input and output (Li *et al.*, 2019). The context layer forms local feedback to realize the time delay between output and input. The model has a memory function for historical data and updates constantly, so it has a dynamic memory function, that help forecast time series. It has been used in power load and motion trajectory forecasting (W. Min *et al.*, 2020; Penghua *et al.*, 2012). The structure of the network is shown in figure 5.9, which has four layers.

The nonlinear state space expression of the network can be expressed as follows (Li *et al.*, 2019):

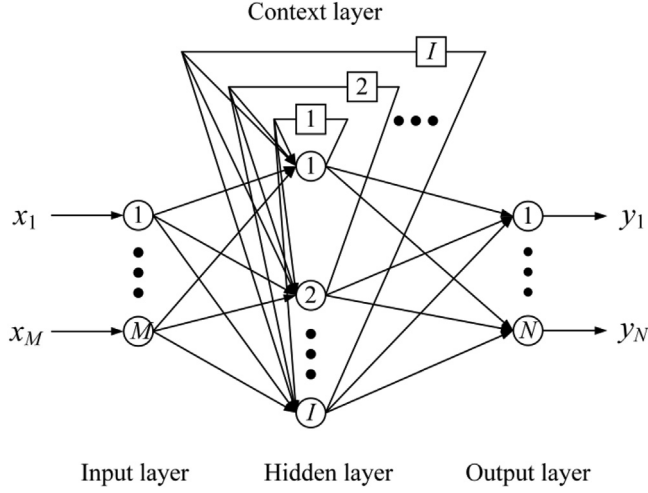


FIG. 5.9 – The structure of Elman neural network.

$$\begin{cases} y(k) = g(w_3 x(k)) \\ x(k) = f(w_1 x_c(k) + w_2 u(k-1)) \\ x_c(k) = x(k-1) \end{cases} \quad (5.5)$$

where y is output, x is the output of the hidden layer, x_c is feedback states, u is the input of the neural network, w_3 is connection weight between the hidden layer and output layer, w_2 is connection weight between the input layer and hidden layer, w_1 is connection weight between the context layer and hidden layer, $g(*)$ is the transfer function of the output layer, $f(*)$ is the transfer function of the hidden layer.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. Then a mapping from 5 sample data points is created to forecast value $x(t)$ by training data sets $[x(t-5), \dots, x(t-2), x(t-1)]$. This means five successive known time-series values are used to forecast the next value. The original vehicle trajectory time series are shown in figure 5.10.

B. Training

The number of hidden layer neurons is 50, and the maximum training iteration of the network is set to be 50. The loss during training is shown in figure 5.11. The training of the Elman neural network converges.

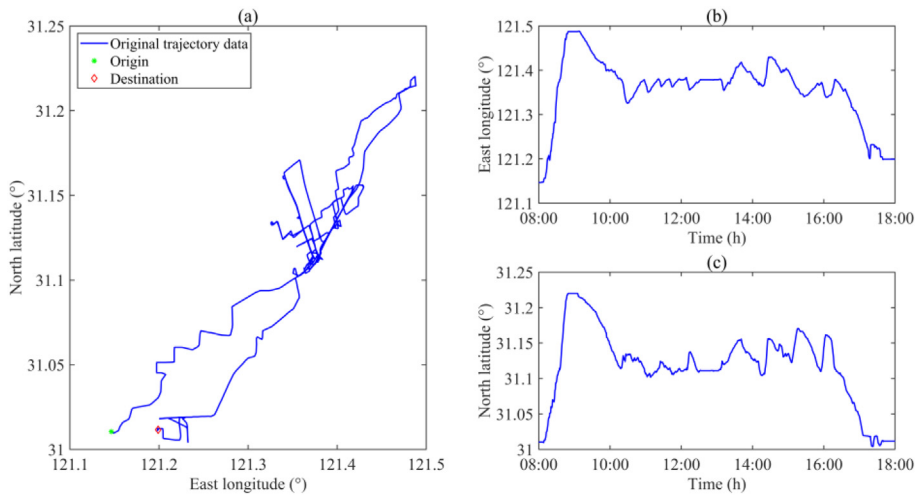


FIG. 5.10 – The original trajectory time series for Elman neural network experiments: (a) trajectory series, (b) east longitude series, (c) north latitude series.

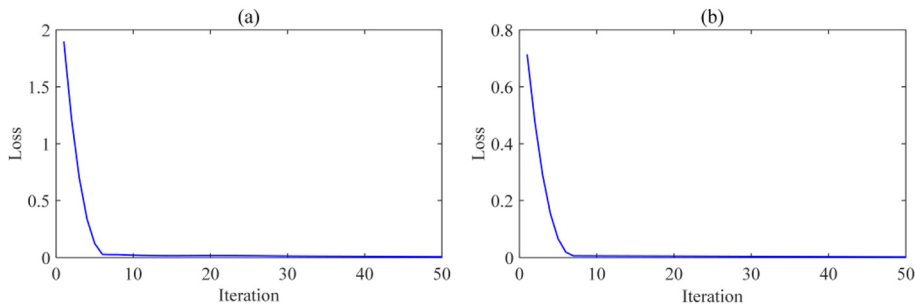


FIG. 5.11 – The loss during the training of the Elman neural network: (a) east longitude series, (b) north latitude series.

C. Forecasting

The trained Elman model is applied to generate forecasting results in testing data.

(3) Forecasting results

The performance of the model is shown in figure 5.12.

The evaluation indices of corresponding forecasting results are shown in table 5.3.

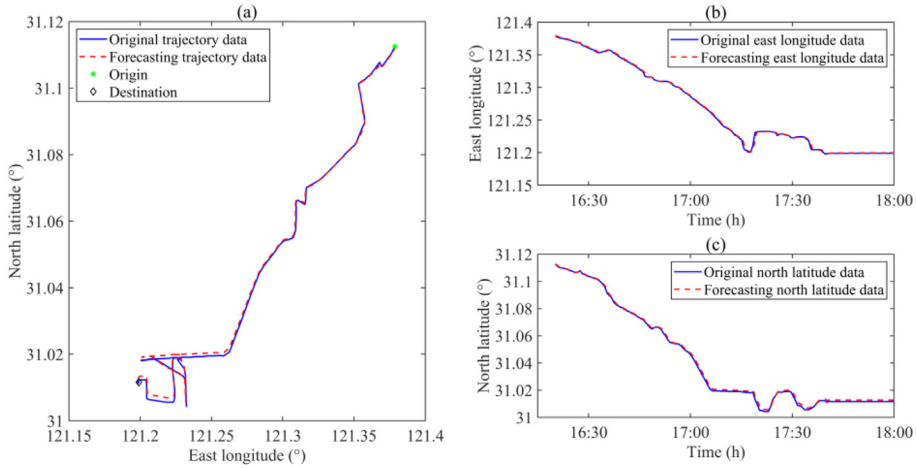


FIG. 5.12 – The forecasting results of the Elman neural network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

TAB. 5.3 – The evaluation indices of the network.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0010	0.0008	0.0015
North latitude series	0.0010	0.0031	0.0011

5.5.2 NAR Neural Network Forecasting Algorithm

(1) Theoretical basis

The NAR neural network is the ensemble of nonlinear autoregressive models and neural networks. The idea of the former is to use its variable as the regression variable, and the linear combination of random variables is used to describe the random variable at the next time in the sequence. The structure of the NAR is shown in figure 5.13, which is composed of input layer, hidden layer, output layer, and input delay (Malekijavan *et al.*, 2019). The model expression of the NAR can be defined as follows (Malekijavan *et al.*, 2019):

$$y(t) = f(y(t-1), y(t-2), y(t-3), \dots, y(t-M)) \quad (5.6)$$

where $y(t)$ is the output at time t , $f(*)$ is the model function, $y(t-1), \dots, y(t-M)$ is historical output at time $t-1, \dots, t-M$, M is delay order.

The NAR neural network includes two network output modes, parallel and series-parallel. In parallel mode, the output value is fed back to the input, and the learning is continued in the hidden layer again. In series-parallel mode, the predicted value is fed back to the input to generate new input data.

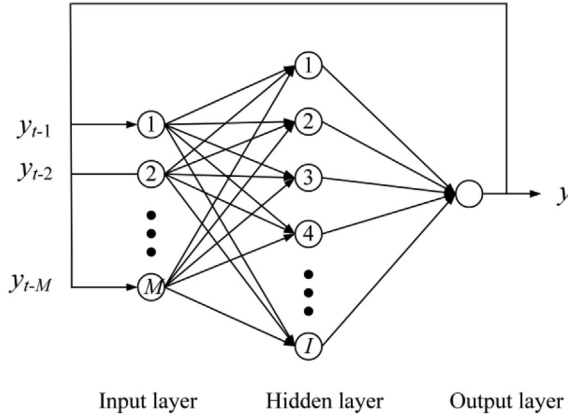


FIG. 5.13 – The structure of the NAR neural network.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series are shown in figure 5.10 in section 5.5.1. Then a mapping from 15 sample data points is created to forecast value $x(t)$ by training data sets $[x(t-15), \dots, x(t-2), x(t-1)]$. This means fifteen successive known time-series values are used to forecast the next value.

B. Training

The number of hidden layer neurons is 10. The loss during training is shown in figure 5.14. The training of the NAR converges.

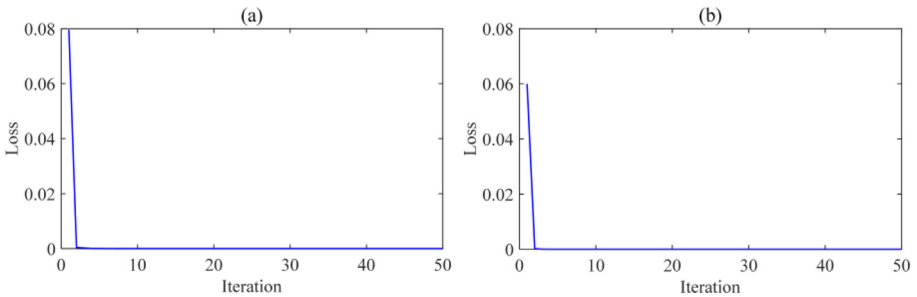


FIG. 5.14 – The loss during the training of the NAR neural network: (a) east longitude series, (b) north latitude series.

C. Forecasting

The trained NAR model is applied to generate forecasting results in testing data.

(3) *Forecasting results*

The performance of the NAR neural network is shown in figure 5.15.

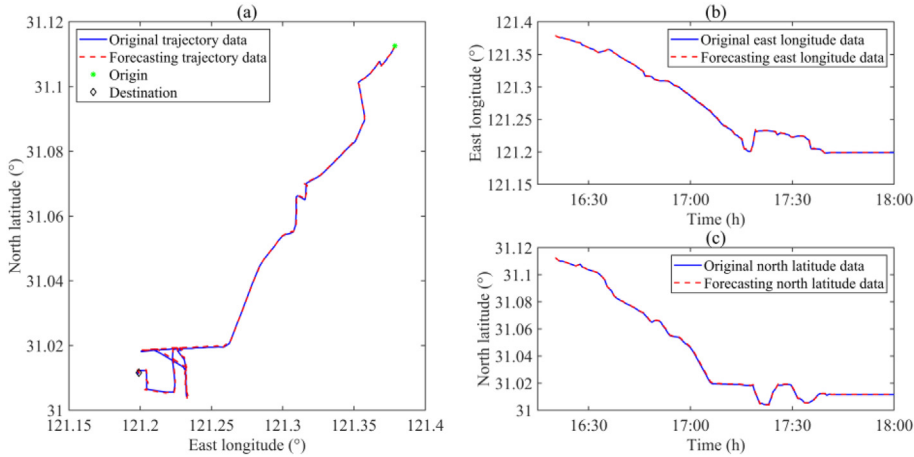


FIG. 5.15 – The forecasting results of the NAR neural network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

The evaluation indices of corresponding forecasting results are shown in table 5.4.

TAB. 5.4 – The evaluation indices of the NAR.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0003	0.0003	0.0005
North latitude series	0.0009	0.0031	0.0011

5.5.3 ANFIS Neural Network Forecasting Algorithm

(1) *Theoretical basis*

A fuzzy neural network combines the learning ability of an artificial neural network and the inference ability of fuzzy control. The ANFIS network is a neural network that uses fuzzy control to strengthen the reasoning ability of the neural network while also increasing the accuracy of the model (Dinh *et al.*, 2014). The advantage of the ANFIS is that the selection of membership function, the calculation of weights, and the formulation of fuzzy rules can all be generated by the learning ability of the neural network. Figure 5.16 shows that the structure of the ANFIS is composed of 5 layers (Yun *et al.*, 2008). The nodes of layer 1 and layer 4 are adaptive, and the node parameters can be adjusted by learning to optimize the

approximation ability of the network. The ANFIS network has two inputs, one output, and three rules in the rule base as follows (Yun *et al.*, 2008):

Rule 1: if x_1 is A_1 , x_2 is B_1 , then $f_1 = p_1x_1 + q_1x_2 + r_1$;

Rule 2: if x_1 is A_2 , x_2 is B_2 , then $f_2 = p_2x_1 + q_2x_2 + r_2$;

Rule 3: if x_1 is A_3 , x_2 is B_3 , then $f_3 = p_3x_1 + q_3x_2 + r_3$;

where x_1, x_2 are inputs, A_i, B_i are linguistic variables, f_i is the output of the fuzzy model, $[p_i, q_i, r_i]$ are consequent parameters.

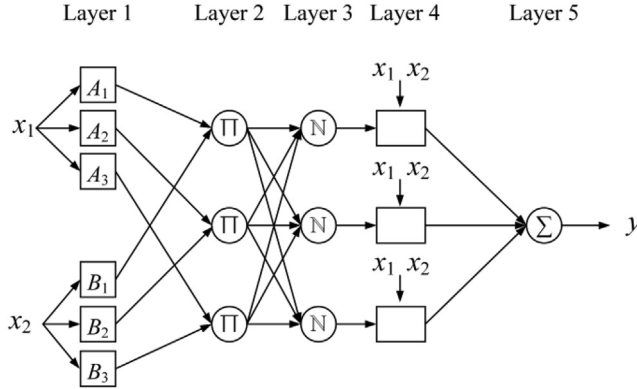


FIG. 5.16 – The structure of the ANFIS network.

Layer 1 fuzzifies inputs and outputs the membership of the corresponding fuzzy set. Each node is an adaptive node and the node function is as follows:

$$\begin{cases} O_{1,i} = \mu_{A_i}(x_1), i = 1, 2, 3 \\ O_{1,j} = \mu_{B_j}(x_2), j = 1, 2, 3 \end{cases} \quad (5.7)$$

where $O(*)$ is membership, $\mu(*)$ is membership function.

Layer 2 calculates fuzzy sets in layer 1, each node is an inactive node and represents a fuzzy rule. The output is defined as follows:

$$O_{2,i} = \mu_{A_i}(x_1)\mu_{B_j}(x_2) \quad (5.8)$$

Layer 3 normalizes the weight of each rule, the number of nodes is the same as layer 2 and each node is an inactive node. The output of layer 3 is as follows:

$$O_{3,i} = \bar{\omega}_i = \frac{\omega_i}{\omega_1 + \omega_2} \quad (5.9)$$

where ω_i is the weight of rules in layer 2.

Layer 4 calculates the outputs of fuzzy rules, each node is an adaptive node and the output of layer 4 is as follows:

$$O_{4,i} = \bar{\omega}_i(p_i x_1 + q_i x_2 + r_i) \quad (5.10)$$

Layer 5 performs de-obfuscation and calculates the system output as follows:

$$O_{5,i} = \sum_i \bar{w}_i f_i \quad (5.11)$$

The learning algorithm of the ANFIS network combines the gradient descent method and the least square method. In forward propagation, the parameters of each node in layer 1 are fixed. According to the initial parameters, the system outputs the result and then uses the least square method to optimize the node parameters of layer 5. The error between the actual value and ideal value is transmitted back to layer 1, and layer 1 uses the gradient descent method to optimize the nonlinear parameters in this layer.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series are shown in figure 5.10 in section 5.5.1. Then a mapping from 4 sample data points is created to forecast value $x(t)$ by training data sets $[x(t-4), x(t-3), x(t-2), x(t-1)]$. This means four successive known time-series values are used to forecast the next value.

B. Building fuzzy inference system

A type-1 model is constructed with four inputs, each input has three generalized bell membership functions. The structure of the initial model is shown in figure 5.17.

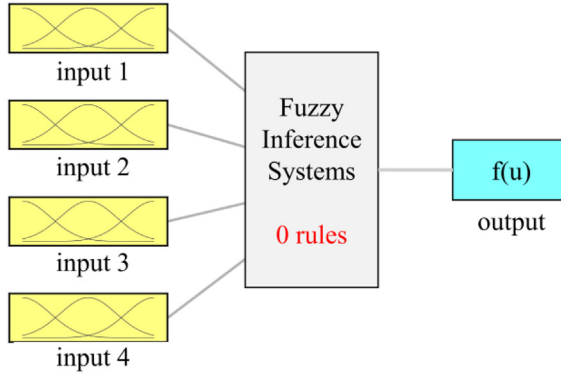


FIG. 5.17 – The structure of the initial type-1 fuzzy inference system.

C. Training

The maximum training iteration of the ANFIS network is set to be 20. The ANFIS network has 193 nodes, 405 linear parameters, 36 nonlinear parameters, and 81 fuzzy rules. The structure of the trained network is shown in figure 5.18.

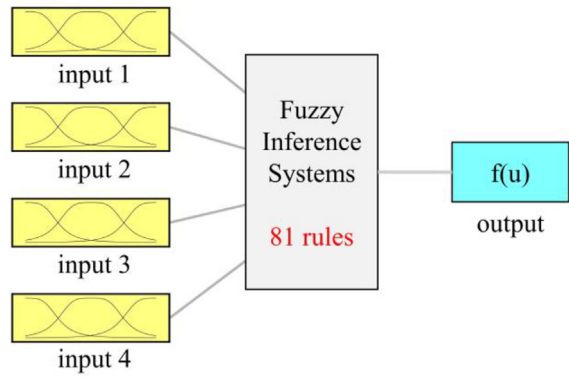


FIG. 5.18 – The structure of the trained type-1 fuzzy inference system.

D. *Forecasting*

The trained ANFIS network is applied to generate forecasting results in testing data.

(3) *Forecasting results*

The performance of the ANFIS network is shown in figure 5.19.

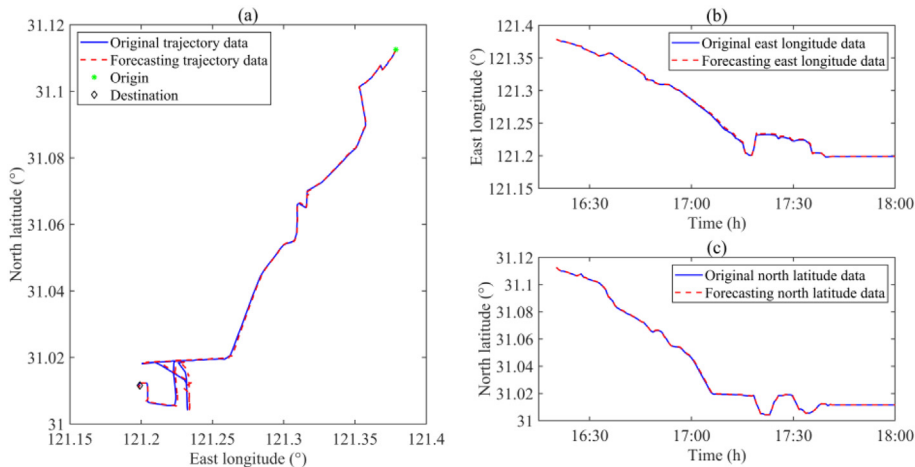


FIG. 5.19 – The forecasting results of the ANFIS network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

The evaluation indices of corresponding forecasting results are shown in table 5.5.

TAB. 5.5 – The evaluation indices of the ANFIS network.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0007	0.0006	0.0011
North latitude series	0.0002	0.0006	0.0003

5.6 Unmanned Vehicle Time Series Deep Learning Forecasting Method

The deep learning methods can extract and learn more in-depth information about time series, which have better nonlinear fitting performance. In this section, three widely used deep learning methods are selected to forecast vehicle trajectory time series, including RNN, LSTM, and Gated Recurrent Unit (GRU).

5.6.1 RNN Deep Neural Network Forecasting Algorithm

(1) Theoretical basis

The RNN deep neural network has the advantage of solving input data correlation with the introduction of time sequence and recursion. The RNN network is characterized by the interconnection of nodes in the same layer and the error feedback mechanism in the hidden layer. That means the output of the hidden layer includes converting information from the hidden layer at the previous time, the RNN network can be used in sequence data forecasting with short-term memory (Du *et al.*, 2020; Tokuyama *et al.*, 2018). The structure of the RNN is shown in figure 5.20, and the hidden layer information of the current time is propagated to the next time (K. Min *et al.*, 2019).

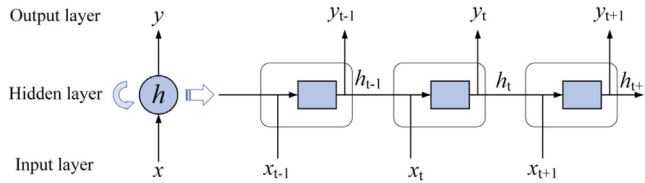


FIG. 5.20 – The structure of the RNN deep neural network.

The signal is brought into recursion when it is transmitted to the next neuron, and it continues in the subsequent signal calculations. The input in the hidden layer includes both the output of the previous hidden layer and the output of the same layer at the previous time. The output of the RNN network can be defined as follows (K. Min *et al.*, 2019):

$$\begin{cases} y_t = f(W_{hy}h_t) \\ h_t = g(W_{xh}x_t + W_{hh}h_{t-1}) \end{cases} \quad (5.12)$$

where y_t is network output at the time t , $f(*)$ is the activation function of the output layer, W_{hy} is the weight matrix between the hidden layer and output layer, h is hidden layer information, $g(*)$ is the activation function of the hidden layer, W_{xh} is weight matrix between the input layer and hidden layer, x_t is network input at time t , W_{hh} is weight matrix between the hidden layer and next hidden layer.

The backpropagation of the RNN network uses the time backpropagation algorithm, which adjusts the weight and threshold of each neuron through error backpropagation and regression of each layer, with the minimum loss function as the optimization goal.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. Then a mapping from 4 sample data points is created to forecast value $x(t)$ by training data sets $[x(t-4), x(t-3), x(t-2), x(t-1)]$. This means four successive known time-series values are used to forecast the next value. The original vehicle trajectory time series is shown in figure 5.21.

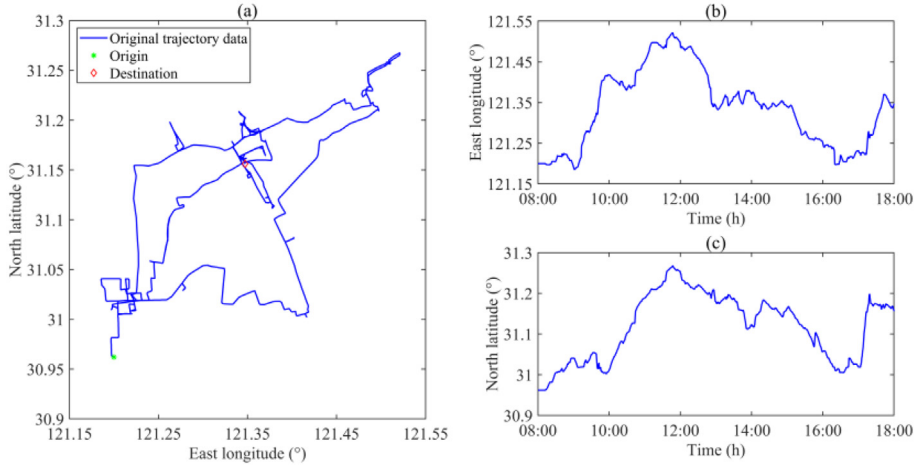


FIG. 5.21 – The original trajectory time series for RNN neural network experiments: (a) trajectory series, (b) east longitude series, (c) north latitude series.

B. Training

The basic network structure is the Elman network with a connection layer to use the hidden layer state at the previous time and network input as hidden layer input, the number of hidden layer neurons is 20, and the maximum training iteration is set

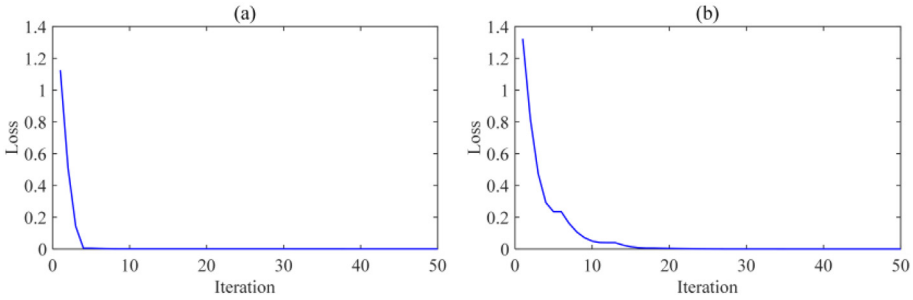


FIG. 5.22 – The loss during the training of the RNN deep neural network: (a) east longitude series, (b) north latitude series.

to be 50. The loss during training is shown in figure 5.22. The training of the RNN converges.

C. Forecasting

The trained RNN model is applied to generate forecasting results in testing data.

(3) Forecasting results

The performance of the RNN deep neural network is shown in figure 5.23. The evaluation indices of corresponding forecasting results are shown in table 5.6.

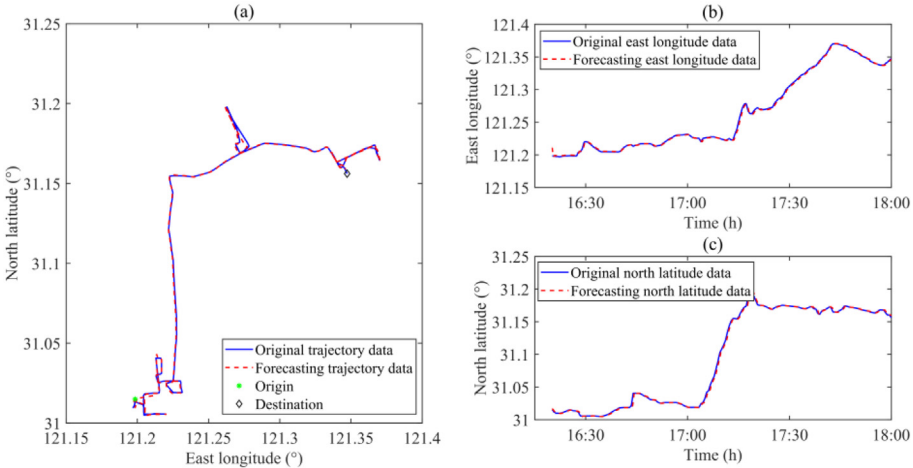


FIG. 5.23 – The forecasting results of the RNN deep neural network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

TAB. 5.6 – The evaluation indices of the RNN.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0012	0.0010	0.0017
North latitude series	0.0012	0.0038	0.0021

5.6.2 LSTM Deep Neural Network Forecasting Algorithm

(1) Theoretical basis

The RNN deep neural network uses the backpropagation algorithm in the optimization of model parameters. The time steps involved in the backpropagation cover all previous times, which involves huge calculations leading to gradient exploding and gradient vanishing problems. To improve the problems, the LSTM was proposed, the characteristic of the LSTM is to retain or update sequence information through the gate structure (Hochreiter and Schmidhuber, 1997). The LSTM network has the same chain structure as the RNN, the structure of the network unit is shown in figure 5.24.

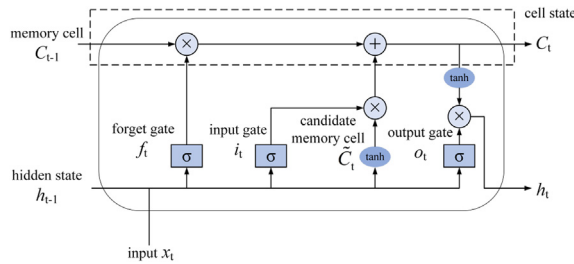


FIG. 5.24 – The structure of the LSTM deep neural network unit.

The forget gate determines the retention or discarding of information, the input of the forget gate is the hidden state h_{t-1} at time $t-1$ and model input x_t at the time t , and the output of the forget gate is a number between 0 and 1, which means the degree of forgetting. The degree of forgetting f_t can be expressed as follows (Althé and de La Fortelle, 2017):

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t]) \quad (5.13)$$

where σ is the sigmoid function, W_f is the weight matrix in forget gate.

The input gate updates the current unit state, which includes the hidden state at the previous time and model input at the current time. Firstly, the information update is determined by i_t , then a candidate memory cell $\tilde{C}_t$ is created to update the

current cell state C_t , which can be expressed as follows (Alth   and de La Fortelle, 2017):

$$\begin{cases} i_t = \sigma(W_i \cdot [h_{t-1}, x_t]) \\ \tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t]) \\ C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \end{cases} \quad (5.14)$$

The output gate determines the current hidden layer state h_t as unit output, which can be expressed as follows (Alth   and de La Fortelle, 2017):

$$\begin{cases} o_t = \sigma(W_o \cdot [h_{t-1}, x_t]) \\ h_t = o_t * \tanh(C_t) \end{cases} \quad (5.15)$$

The LSTM network can control unit state by these gates, which can save the needed information and forget the less important information.

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series is shown in figure 5.21 in section 5.6.1.

B. Training

The number of hidden layer neurons is 100, the maximum training iteration is set to 50. The loss during training is shown in figure 5.25. The training of the LSTM converges.

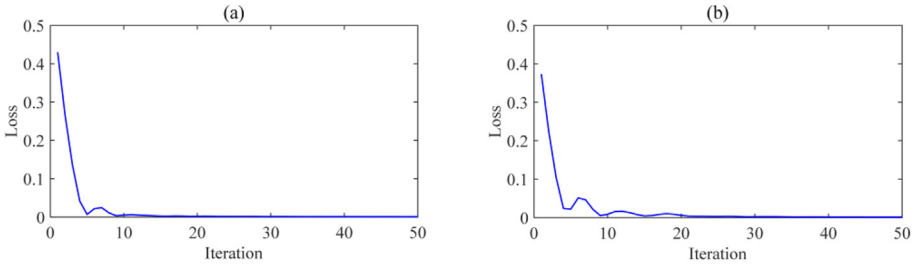


FIG. 5.25 – The loss during the training of the LSTM deep neural network: (a) east longitude series, (b) north latitude series.

C. Forecasting

The trained LSTM model is applied to generate forecasting results in testing data.

(3) Forecasting results

The performance of the LSTM deep neural network is shown in figure 5.26.

The evaluation indices of corresponding forecasting results are shown in table 5.7.

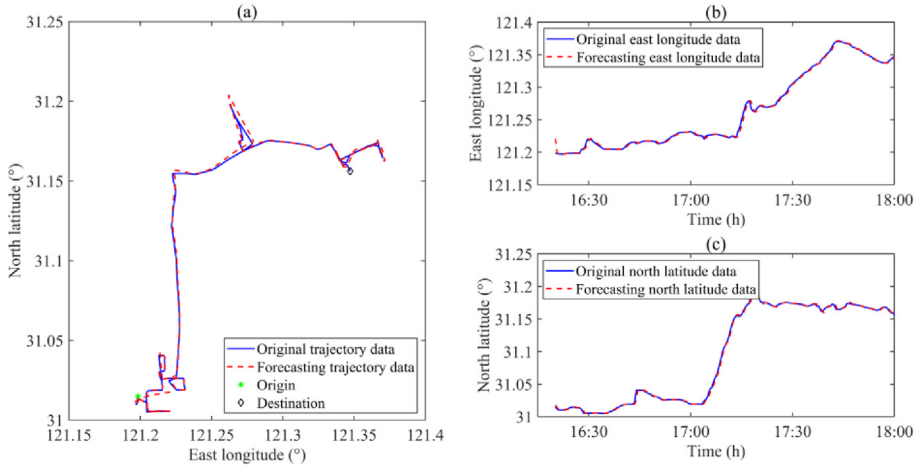


FIG. 5.26 – The forecasting results of the LSTM deep neural network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

TAB. 5.7 – The evaluation indices of the LSTM.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0015	0.0012	0.0024
North latitude series	0.0012	0.0039	0.0021

5.6.3 GRU Deep Neural Network Forecasting Algorithm

(1) Theoretical basis

The LSTM deep neural network was proposed to avoid gradient exploding and gradient vanishing problems, but the LSTM model has a complex structure leading to a long training time. The GRU deep neural network was proposed with a simplified structure (Cho *et al.*, 2014). The GRU network also has the same chain structure as the RNN, the structure of the network unit is shown in figure 5.27.

The reset gate determines the degree to which the state information at the previous time is written to the candidate's hidden state $\tilde{h}_t$. The input of the reset gate is the hidden state h_{t-1} at time $t-1$ and model input x_t at the time t , the output r_t of the reset gate is a number between 0 and 1, which can be defined as follows (Cho *et al.*, 2014):

$$\begin{cases} r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \\ \tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t * h_{t-1}, x_t]) \end{cases} \quad (5.16)$$

where σ is the sigmoid function, W_r is the weight matrix in the reset gate.

The update gate determines the degree to which the state information at the previous time is retained in the current state. The input of the update gate is the

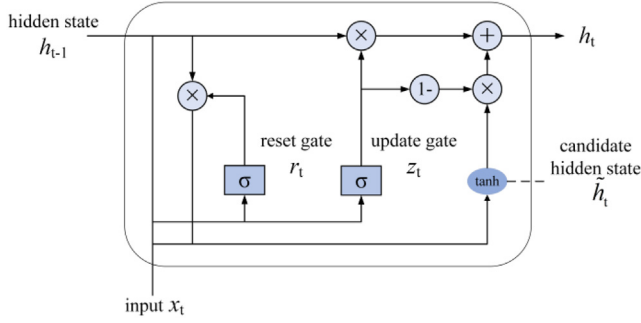


FIG. 5.27 – The structure of the GRU deep neural network unit.

same as the reset gate, and the output z_t of the update gate is a number between 0 and 1, which can be expressed as follows (Cho *et al.*, 2014):

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad (5.17)$$

The output of the GRU unit is the hidden state h_t at time t and model output y_t , which can be expressed as follows (Cho *et al.*, 2014):

$$\begin{cases} h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \\ y_t = \sigma(W_o \cdot h_t) \end{cases} \quad (5.18)$$

(2) Modeling steps

A. Data preprocessing

The modeling data is processed as section 5.4.1, which contains 3600 data, the 1st–3000th is training data, and the 3001st–3600th is testing data. The original vehicle trajectory time series is shown in figure 5.21 in section 5.6.1.

B. Training

The number of hidden layer neurons is 100, the maximum training iteration is set to 50. The loss during training is shown in figure 5.28. The training of the GRU model converges.

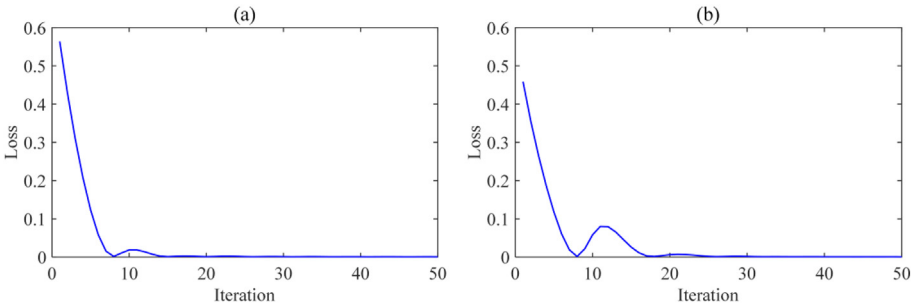


FIG. 5.28 – The loss during the training of the GRU deep neural network: (a) east longitude series, (b) north latitude series.

C. Forecasting

The trained GRU model is applied to generate forecasting results in testing data.

(3) Forecasting results

The performance of the GRU deep neural network is shown in figure 5.29.

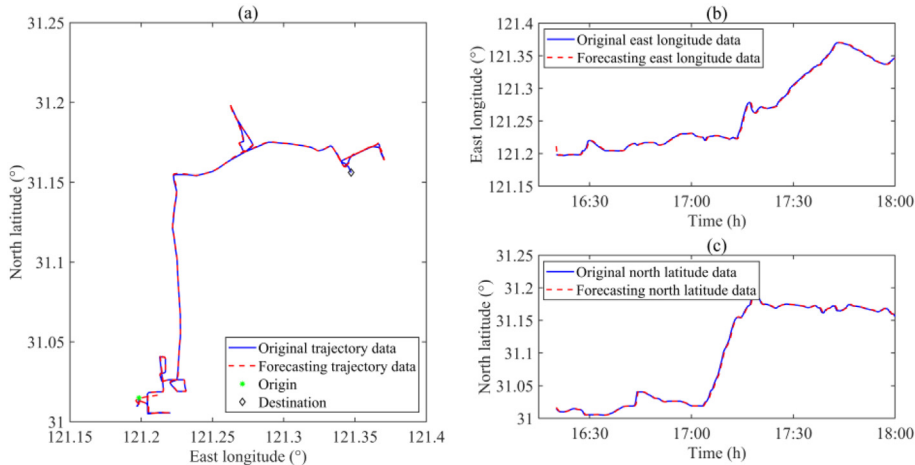


FIG. 5.29 – The forecasting results of the GRU deep neural network: (a) trajectory series, (b) east longitude series, (c) north latitude series.

The evaluation indices of corresponding forecasting results are shown in table 5.8.

TAB. 5.8 – The evaluation indices of the GRU.

Series	Evaluation indices		
	MAE (°)	MAPE (%)	RMSE (°)
East longitude series	0.0010	0.0008	0.0015
North latitude series	0.0009	0.0029	0.0016

5.7 Comparative Analysis of Forecasting Performance

The performance of the model and evaluation indices of different models are analyzed and compared in the section.

5.7.1 Analysis of Statistical Methods

According to the results of statistical methods in section 5.4, the following conclusions can be made:

- (a) From figures 5.5 and 5.8, the statistical models can only fit the basic trend of the original vehicle trajectory time series, but cannot forecast accurately,

including the Kalman filter and fuzzy time series forecasting algorithm. The Kalman filter is only suitable for linear systems and is easily affected by system noise and measurement noise.

- (b) According to tables 5.1 and 5.2, the evaluation indices show that the Kalman filter model and fuzzy time series model have a good performance. This phenomenon can be explained as follows: original vehicle trajectory time series varies little from 121.25° to 121.5° in east longitude and 30.9° to 31.3° in north latitude, the trajectory data changing range is only 0.25° and 0.4° , which makes the value of all evaluation indices low.

5.7.2 Analysis of Intelligent Methods

According to the results of intelligent methods in section 5.5, the following conclusions can be made:

From figures 5.12, 5.15, and 5.19, the intelligent models can forecast the vehicle trajectory series accurately, including the Elman, NAR, and ANFIS forecasting algorithms. The evaluation indices in tables 5.3–5.5, the evaluation indices show that the ANFIS model has the highest prediction accuracy, the MAE, MAPE, and RMSE of it are 0.0007° , 0.0006% and 0.0011° in east longitude series, 0.0002° , 0.0006% and 0.0003° in north latitude series, respectively. Conclusively, the NAR model is proven to be an effective and outstanding vehicle trajectory time series forecasting model.

5.7.3 Analysis of Deep Learning Methods

According to the results of deep learning methods in section 5.6, the following conclusions can be made:

From figures 5.23, 5.26, and 5.29, the deep learning models can forecast the vehicle trajectory series accurately, including the RNN, LSTM, and GRU forecasting algorithms. According to the evaluation indices in tables 5.6–5.8, the evaluation indices show that the GRU model has the highest prediction accuracy in forecasting, the MAE, MAPE, and RMSE of it are 0.0010° , 0.0008% and 0.0015° in east longitude series, 0.0009° , 0.0029% and 0.0016° in north latitude series, respectively. The performance of deep learning modes is easily affected by the hyperparameter of the neural network, which needs decomposition or optimization.

5.8 Unmanned Vehicle Navigation Control Based on Multi-Source Position Time Series Fusion

5.8.1 Unmanned Vehicle Fusion Positioning

High-precision positioning is a prerequisite for unmanned vehicle decision-making and motion control, the unmanned vehicle can perform path planning and set control parameters only with the location information. According to the types of

sensors and data processing methods, vehicle positioning technologies can be divided into three categories, respectively based on dead reckoning, satellite, and environmental perception.

(1) *Positioning based on dead reckoning*

With the sensor data, the relative posture change between two data frames of the vehicle can be calculated according to vehicle dynamics equations, the vehicle trajectory can be obtained by continuously calculating the movement state of the vehicle. The time between two frames is usually short, the vehicle dynamics models can be divided into linear motion models and curvilinear motion models with approximate assumptions (Schubert *et al.*, 2011). The vehicle only goes straight in linear models, which consist of the constant velocity model and constant acceleration model. The vehicle turns with a constant turn rate in curvilinear models, which include the constant turn rate and velocity model and the constant turn rate and acceleration model.

The commonly used sensors of dead reckoning are the encoder and the IMU (El-Mowafy and Kubo, 2017; Pratama *et al.*, 2016). The positioning based on dead reckoning is a relative positioning, which has good stability and accuracy in a short time. The positioning accuracy depends on the accuracy of sensors and pose estimation at the previous time, and the error will accumulate over time.

(2) *Positioning based on satellite*

The vehicle position and speed can be obtained by resolving satellite signals from the GNSS. The GNSS can achieve global positioning, but its positioning accuracy in the motion state is insufficient. With this problem, Real-Time Kinematic (RTK) measurement was proposed to achieve centimeter-level positioning accuracy (Takanose *et al.*, 2021). The RTK-GNSS can achieve high-precision positioning, but satellite signals are easily affected by the environment, especially in urban three-dimensional traffic.

(3) *Positioning based on environmental perception*

Environmental perception of unmanned vehicles mainly uses cameras and lidar, the positioning-based environmental perception can be divided into positioning based on beacon detection, positioning based on scene matching, and positioning based on visual odometry.

Vehicle position estimation can be achieved by using the triangulation method for three or more beacons in positioning based on scene matching. If the global coordinates of beacons are known, they can be recognized by onboard sensors, such as traffic lights, road signs, etc. (Wu and Ranganathan, 2013). The local scene is constructed by the perceived local feature information of the surrounding environment, then the constructed scene is compared with the stored global environment scene map. Once the map matching is achieved, the vehicle position is determined (Zhou *et al.*, 2019). Visual odometry uses images to estimate vehicle motion information without prior information about scenes and motions (Schreiber *et al.*, 2016).

The unmanned vehicle positioning environment is complex and dynamic, and a single sensor has its shortcomings. It is necessary to use multi-sensor fusion

technology to achieve more accurate and reliable positioning functions. The improvement of positioning accuracy and robustness depends on the maximum extraction of complementary and redundant information between each sensor. The fusion positioning of unmanned vehicles is shown in figure 5.30, which is integrated by multiple positioning methods.

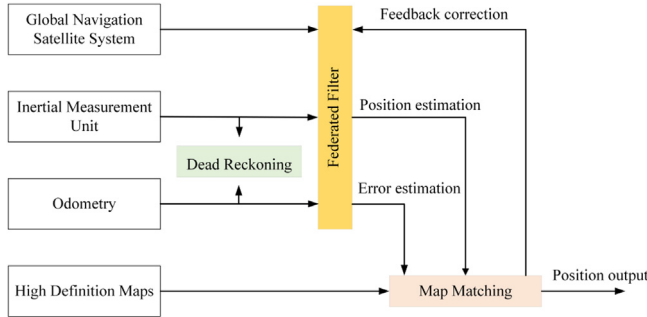


FIG. 5.30 – The fusion positioning of unmanned vehicle.

The fusion structure of multi-sensor fusion consists of pixel-level fusion, feature-level fusion, and decision-level fusion, which is shown in figure 5.31 (Wan *et al.*, 2018). The commonly used sensor fusion technologies include reasoning and statistical methods, Bayesian inference, fuzzy set theory, and neural networks.

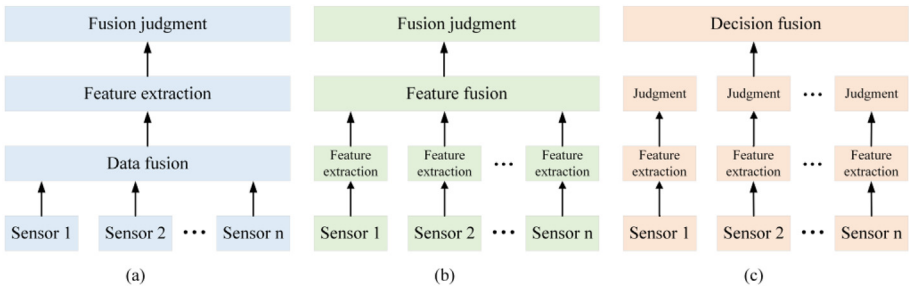


FIG. 5.31 – The structure of multi-sensor fusion: (a) pixel level fusion, (b) feature level fusion, (c) decision level fusion.

5.8.2 Unmanned Vehicle Navigation Control

Based on the surrounding environment information and vehicle status information provided by the perception system, the vehicle performs path planning and controls vehicle movement. Path planning is to plan the drivable area and driving guidance for the unmanned vehicle according to driving tasks and real-time environment. Then the vehicle is controlled to follow the planned path accurately and quickly.

Path planning of unmanned vehicles consists of global path planning and local path planning (Ayawli *et al.*, 2018). Global path planning does not take road crossing, width, curvature, etc. into account, and the uncertainty in the vehicle driving process, so local path planning is needed on the basis of local environmental information. Local path planning is to formulate optimal controllable driving path based on the target requirements of unmanned vehicles and perceived local environment information under the guidance of path generated by global path planning. The global path planning algorithms can be divided into grid method, visibility graph method, topological method, neural network, etc. (González *et al.*, 2015). The global path planning algorithms include the artificial potential field method, virtual force field method, vector field histogram method, reinforcement learning, etc. (Sarkar and Mohan, 2019; You *et al.*, 2019).

Unmanned vehicle motion control can be divided into lateral control and longitudinal control (Klomp *et al.*, 2019). Lateral control is to control the unmanned vehicle to run along the road axis and longitudinal control is to control the vehicle to adjust speed.

The vehicle can only move in the direction of the body, which is a typical non-holonomic constraint system and is a complex system with highly nonlinear dynamic characteristics and parameter uncertainties. Many methods have been proposed for vehicle lateral control, such as optimum control, robust control, sliding mode control, fuzzy control, model predictive control, neural network, deep reinforcement learning, etc. (Wasala *et al.*, 2020; Ni *et al.*, 2016; Wang *et al.*, 2015). Vehicle longitudinal control can be divided into direct control and hierarchical control. The direct control structure controls all subsystems by a longitudinal controller. The hierarchical control structure designs upper and lower controllers to make the system simple. Some coordinated control algorithms based on non-convex optimization and neural network for vehicle lateral control and longitudinal control are also proposed (Z. Sun *et al.*, 2020; Luo *et al.*, 2015).

5.9 Unmanned Vehicle Charging Control Based on Multi-Source Power Time Series Fusion

The power management of unmanned vehicles is a security guarantee for the battery, which is controlled by the battery management system, and the structure is shown in figure 5.32.

The common power management strategies include rule-based power management and optimization-based power management. The former include static logic threshold control and fuzzy control, which mainly involve rules such as system operation mode, power distribution, and actuator state switching (Denis *et al.*, 2015). The optimization-based power management includes global optimal power management and instantaneous optimal power management. Vehicle power management is regarded as a nonlinear optimal solution problem with multi-dimensional physical constraints, which are composed of physical constraints of control variables, constraints of state variables, and cost function (Chen *et al.*, 2019; N. Kim *et al.*, 2010).

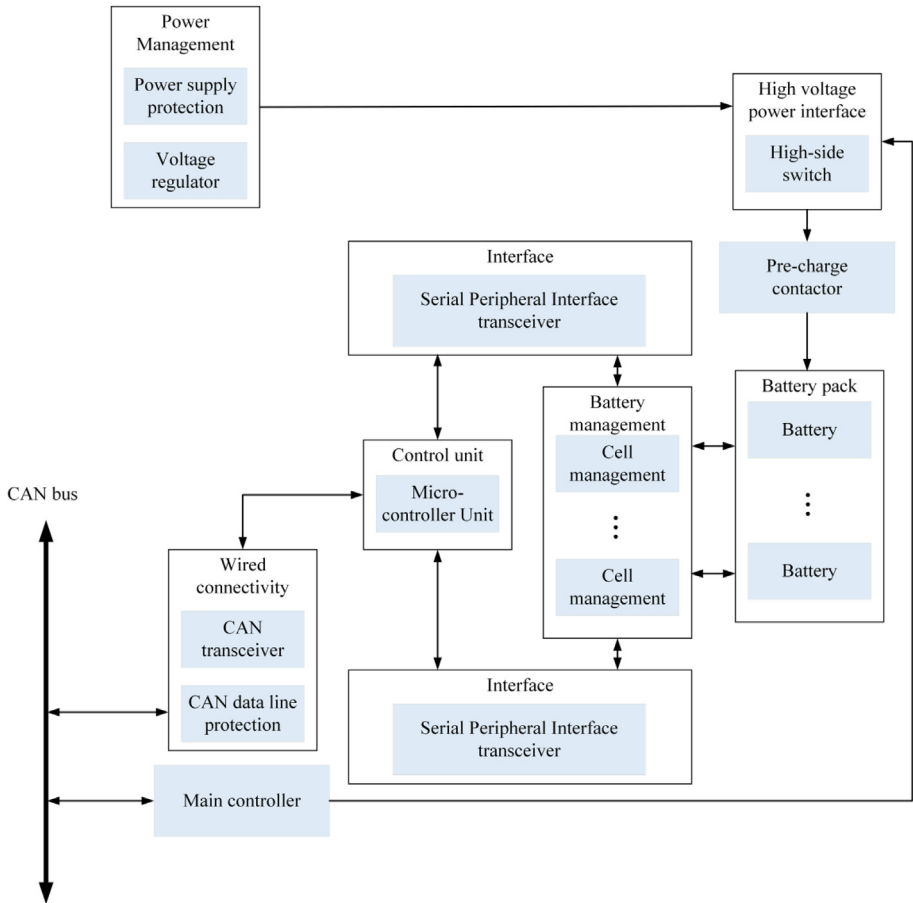


FIG. 5.32 – The structure of the battery management system.

In addition, vehicle speed forecasting, road gradient forecasting, and driver behavior forecasting are integrated into unmanned vehicle power management to improve the independence of vehicle power management (Malikopoulos, 2015; C. Sun *et al.*, 2014).

5.10 Conclusions

Positioning and trajectory forecasting are essential for unmanned vehicles, which can help vehicles obtain the optimal planned path and improve the efficiency of interaction with surrounding traffic participants. In this chapter, vehicle trajectory time series modeling and forecasting methods are introduced in detail. Different statistical models, intelligent models, and deep learning models are built for vehicle trajectory time series forecasting. Through the vehicle trajectory time series modeling and prediction, unmanned vehicle navigation control and charging control can be achieved.

Combing the results of different methods, the following conclusions of vehicle trajectory time series forecasting can be drawn:

- (a) In the two statistical models, the Kalman filter forecasting model has good performance in evaluation indices. However, the method is easily influenced by noise and does not have sufficient stability. The accuracy performance of the fuzzy series time method cannot meet the requirements of practical applications.
- (b) Three intelligent models can forecast the vehicle trajectory time series effectively. Among them, the ANFIS model is proven to be the best intelligent forecasting model that can integrate the inference ability of fuzzy control and the learning ability of neural networks.
- (c) Deep learning methods can extract deeper information from time series. Three deep learning methods in this chapter have effective performance in vehicle trajectory time series forecasting. The GRU model has the best performance with a simple structure and less calculating time compared with RNN and LSTM models.

References

- Abbas M. T., Jibran M. A., Afaq M., Song W. (2020) An adaptive approach to vehicle trajectory prediction using multimodel Kalman filter, *Trans. Emerg. Telecommun. Technol.* **31**(5), e3734.
- Althché F., de La Fortelle A. (2017) *An LSTM network for highway trajectory prediction*. Paper presented at the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).
- Althoff M., Mergel A. (2011) Comparison of Markov chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars, *IEEE Trans. Intell. Transp. Syst.* **12**(4), 1237–1247.
- Ayawli B. B. K., Chellali R., Appiah A. Y., Kyeremeh F. (2018) An overview of nature-inspired, conventional, and hybrid methods of autonomous vehicle path planning, *J. Adv. Transp.* **2018**, 8269698.
- Baek M., Jeong D., Choi D., Lee S. (2020) Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications, *Sensors* **20**(1), 288.
- Bahram M., Hubmann C., Lawitzky A., Aeberhard M., Wollherr D. (2016) A combined model-and learning-based framework for interaction-aware maneuver prediction, *IEEE Trans. Intell. Transp. Syst.* **17**(6), 1538–1550.
- Bose M., Mali K. (2019) Designing fuzzy time series forecasting models: A survey. *Int. J. Approx. Reason.* **111**, 78–99.
- Chen Z., Shu X., Xiao R., Yan W., Liu Y., Shen J. (2019) Optimal charging strategy design for lithium-ion batteries considering minimization of temperature rise and energy loss, *Int. J. Energy Res.* **43**(9), 4344–4358.
- Cheng Y., Li S. (2015) *Implementation of prediction of ball position in robot soccer using Kalman Filter*. Paper presented at the Proceedings of the 2015 6th International Conference on Manufacturing Science and Engineering.
- Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Denis N., Dubois M. R., Desrochers A. (2015) Fuzzy-based blended control for the energy management of a parallel plug-in hybrid electric vehicle, *IET Intell. Transp. Syst.* **9**(1), 30–37.
- Dinh H. T., Kamalapurkar R., Bhasin S., Dixon W. E. (2014) Dynamic neural network-based robust observers for uncertain nonlinear systems, *Neural Netw.* **60**, 44–52.

- Dong L., Wang P., Yan F. (2019) Damage forecasting based on multi-factor fuzzy time series and cloud model, *J. Intell. Manuf.* **30**(2), 521–538.
- Du Y., Cui N., Li H., Nie H., Shi Y., Wang M., Li T. (2020) *The Vehicle's Velocity Prediction Methods Based on RNN and LSTM Neural Network*. Paper presented at the 2020 Chinese Control And Decision Conference (CCDC).
- El-Mowafy A., Kubo N. (2017) Integrity monitoring of vehicle positioning in urban environment using RTK-GNSS, IMU and speedometer, *Meas. Sci. Technol.* **28**(5), 055102.
- Farzaneh Y., Akbarzadeh A., Akbari A. A. (2014) Online bio-inspired trajectory generation of seven-link biped robot based on T-S fuzzy system, *Appl. Soft Comput.* **14**, 167–180.
- Gindele T., Brechtel S., Dillmann R. (2015) Learning driver behavior models from traffic observations for decision making and planning, *IEEE Intell. Transp. Syst. Mag.* **7**(1), 69–79.
- González D., Pérez J., Milanés V., Nashashibi F. (2015) A review of motion planning techniques for automated vehicles, *IEEE Trans. Intell. Transp. Syst.* **17**(4), 1135–1145.
- Greenberg J. N., Tan X. (2020) Dynamic optical localization of a mobile robot using Kalman filtering-based position prediction, *IEEE/ASME Trans. Mechatron.* **25**(5), 2483–2492.
- Hochreiter S., Schmidhuber J. (1997) Long short-term memory, *Neural Comput.* **9**(8), 1735–1780.
- Jeong Y., Yi K. (2019) Target vehicle motion prediction-based motion planning framework for autonomous driving in uncontrolled intersections, *IEEE Trans. Intell. Transp. Syst.* **22**(1), 168–177.
- Kasper D., Weidl G., Dang T., Breuel G., Tamke A., Wedel A., Rosenstiel W. (2012) Object-oriented Bayesian networks for detection of lane change maneuvers, *IEEE Intell. Transp. Syst. Mag.* **4**(3), 19–31.
- Kim B., Kang C. M., Kim J., Lee S. H., Chung C. C., Choi J. W. (2017) *Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network*. Paper presented at the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).
- Kim N., Cha S., Peng H. (2010) Optimal control of hybrid electric vehicles based on Pontryagin's minimum principle, *IEEE Trans. Control Syst. Technol.* **19**(5), 1279–1287.
- Klingschmitt S., Platho M., Groß H.-M., Willert V., Eggert J. (2014) *Combining behavior and situation information for reliably estimating multiple intentions*. Paper presented at the 2014 IEEE Intelligent Vehicles Symposium Proceedings.
- Klomp M., Jonasson M., Laine L., Henderson L., Regolin E., Schumi S. (2019) Trends in vehicle motion control for automated driving on public roads, *Veh. Syst. Dyn.* **57**(7), 1028–1061.
- Kocić J., Jovičić N., Drndarević V. (2018) *Sensors and sensor fusion in autonomous vehicles*. Paper presented at the 2018 26th Telecommunications Forum (TELFOR).
- Kumar P., Perrollaz M., Lefevre S., Laugier C. (2013) *Learning-based approach for online lane change intention prediction*. Paper presented at the 2013 IEEE Intelligent Vehicles Symposium (IV).
- Lee D., Kwon Y. P., McMains S., Hedrick J. K. (2017) *Convolution neural network-based lane change intention prediction of surrounding vehicles for ACC*. Paper presented at the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).
- Lefèvre S., Vasquez D., Laugier C. (2014) A survey on motion prediction and risk assessment for intelligent vehicles, *ROBOMECH J.* **1**(1), 1–14.
- Lenz D., Diehl F., Le M. T., Knoll A. (2017) *Deep neural networks for Markovian interactive scene prediction in highway scenarios*. Paper presented at the 2017 IEEE Intelligent Vehicles Symposium (IV).
- Li C., Wang Y., Zhou Z., Wu J., Jin W., Yin C. (2019) *Lateral state estimation of preceding target vehicle based on multiple neural network ensemble*. Paper presented at the 2019 IEEE Intelligent Vehicles Symposium (IV).
- Liu S., Liu Y., Ni L. M., Fan J., Li M. (2010) *Towards mobility-based clustering*. Paper presented at the Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining.
- Luo Y., Cao K., Xiang Y., Li K. (2015) Vehicle stability and attitude improvement through the coordinated control of longitudinal, lateral and vertical tyre forces for electric vehicles, *Int. J. Veh. Des.* **69**(1–4), 25–49.

- Malekijavan A., Aslinezhad M., Abbasi P. (2019) Position improvement of GPS/INS using NAR neural network and robust Kalman filter during GPS outage, *J. Aeronaut. Eng.* **21**(2), 46–56.
- Malikopoulos A. A. (2015) A multiobjective optimization framework for online stochastic optimal control in hybrid electric vehicles, *IEEE Trans. Control Syst. Technol.* **24**(2), 440–450.
- Min K., Kim D., Park J., Huh K. (2019) RNN-based path prediction of obstacle vehicles with deep ensemble, *IEEE Trans. Veh. Technol.* **68**(10), 10252–10256.
- Min W., Jiawei W., Jinhui G., Lihua S., Bogong A. (2020) Multi-point prediction of aircraft motion trajectory based on GA-Elman-regularization neural network, *Integr. Ferroelectr.* **210**(1), 116–127.
- Mo Y., Yu D., Song J., Zheng K., Guo Y. (2016) Vehicle position updating strategy based on Kalman filter prediction in VANET environment, *Discrete Dyn. Nat. Soc.* **2016**, 1404396.
- Morales J. J., Roysdon P. F., Kassas Z. M. (2016) *Signals of opportunity aided inertial navigation*. Paper presented at the Proceedings of the 29th international technical meeting of the satellite division of the institute of navigation.
- Morris B., Doshi A., Trivedi M. (2011) *Lane change intent prediction for driver assistance: On-road design and evaluation*. Paper presented at the 2011 IEEE Intelligent Vehicles Symposium (IV).
- Ni L., Gupta A., Falcone P., Johannesson L. (2016) Vehicle lateral motion control with performance and safety guarantees, *IFAC-Pap.* **49**(11), 285–290.
- Ortiz M. G., Fritsch J., Kummert F., Gepperth A. (2011) *Behavior prediction at multiple time-scales in inner-city scenarios*. Paper presented at the 2011 IEEE Intelligent Vehicles Symposium (IV).
- Penghua L., Yi C., Qingyu X., Ke Z., Liping C. (2012) *Short-term load forecasting using hybrid quantized elman neural model*. Paper presented at the Proceedings of the 31st Chinese Control Conference.
- Peron D., Giordani M., Zorzi M. (2019) *An efficient requirement-aware attachment policy for future millimeter wave vehicular networks*. Paper presented at the 2019 IEEE Intelligent Vehicles Symposium (IV).
- Phillips D. J., Wheeler T. A., Kochenderfer M. J. (2017) *Generalizable intention prediction of human drivers at intersections*. Paper presented at the 2017 IEEE intelligent vehicles symposium (IV).
- Pratama P. S., Nguyen T. H., Kim H. K., Kim D. H., Kim S. B. (2016) Positioning and obstacle avoidance of automatic guided vehicle in partially known environment, *Int. J. Control, Autom. Syst.* **14**(6), 1572–1581.
- Sarkar S. B., Mohan B. C. (2019) *Review on autonomous vehicle challenges*. Paper presented at the International Conference on Artificial Intelligence and Cognitive Computing.
- Schreiber M., Königshof H., Hellmund A.-M., Stiller C. (2016) *Vehicle localization with tightly coupled GNSS and visual odometry*. Paper presented at the 2016 IEEE Intelligent Vehicles Symposium (IV).
- Schreier M., Willert V., Adamy J. (2014) *Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems*. Paper presented at the 17th international IEEE conference on intelligent transportation systems (ITSC).
- Schubert R., Adam C., Obst M., Mattern N., Leonhardt V., Wanielik G. (2011) *Empirical evaluation of vehicular models for ego motion estimation*. Paper presented at the 2011 IEEE intelligent vehicles symposium (IV).
- Sharma O., Sahoo N. C., Puhan N. (2019) *A Survey on Smooth Path Generation Techniques for Nonholonomic Autonomous Vehicle Systems*. Paper presented at the IECON 2019–45th Annual Conference of the IEEE Industrial Electronics Society.
- Śmieszek M., Dobrzańska M. (2015) Application of Kalman filter in navigation process of automated guided vehicles, *Metrol. Meas. Syst.* **22**(3), 443–454.
- Sun C., Hu X., Moura S. J., Sun F. (2014) Velocity predictors for predictive energy management in hybrid electric vehicles, *IEEE Trans. Control Syst. Technol.* **23**(3), 1197–1204.
- Sun Z., Wang R., Ye Q., Wei Z., Yan B. (2020) Investigation of intelligent vehicle path tracking based on longitudinal and lateral coordinated control, *IEEE Access* **8**, 105031–105046.
- Takanose A., Atsumi Y., Takikawa K., Meguro J. (2021) Improvement of reliability determination performance of real time kinematic solutions using height trajectory, *Sensors* **21**(2), 657.

- Tokuyama Y., Fukushima Y., Yokohira T. (2018) *The effect of using attribute information in network traffic prediction with deep learning*. Paper presented at the 2018 international conference on information and communication technology convergence (ICTC).
- Veeraraghavan H., Papanikolopoulos N., Schrater P. (2006) *Deterministic sampling-based switching Kalman filtering for vehicle tracking*. Paper presented at the 2006 IEEE Intelligent Transportation Systems Conference.
- Wan G., Yang X., Cai R., Li H., Zhou Y., Wang H., Song S. (2018) *Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes*. Paper presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA).
- Wang X., Fu M., Ma H., Yang Y. (2015) Lateral control of autonomous vehicles based on fuzzy logic. *Control Eng. Pract.* **34**, 1–17.
- Wasala A., Byrne D., Miesbauer P., O'Hanlon J., Heraty P., Barry P. (2020) Trajectory based lateral control: a reinforcement learning case study, *Eng. Appl. Artif. Intell.* **94**, 103799.
- Wu T., Ranganathan A. (2013) *Vehicle localization using road markings*. Paper presented at the 2013 IEEE Intelligent Vehicles Symposium (IV).
- You C., Lu J., Filev D., Tsiotras P. (2019) Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning, *Rob. Auton. Syst.* **114**, 1–18.
- Yun Z., Quan Z., Caixin S., Shaolan L., Yuming L., Yang S. (2008) RBF neural network and ANFIS-based short-term load forecasting approach in real-time price environment, *IEEE Trans. Power Syst.* **23**(3), 853–858.
- Zhang S., Zhao X., Lei W., Yu Q., Wang Y. (2020) Front vehicle detection based on multi-sensor fusion for autonomous vehicle, *J. Intell. Fuzzy Syst.* **38**(1), 365–377.
- Zhou J., Zhao X., Cheng X., Xu Z., Zhao H. (2019) Vehicle ego-localization based on streetscape image database under blind area of global positioning system, *J. Shanghai Jiaotong Univ. (Sci.)* **24**(1), 122–129.

Chapter 6

Wearable Assistive Robot Time Series Predictive Control

6.1 Introduction

The wearable assistive robot is a mechatronics system designed for the shape and function of the human body, which can cooperate with people in the right position and attitude (Moreno *et al.*, 2008). Depending on the service provided, the wearable assistive robot can be divided into upper limb type, lower limb type, and whole-body limb type. Depending on the service function, the wearable assistive robot can be divided into repair function type (straightening machine), enhancement function type, and replacement function type (artificial limb).

Because the wearable assistive robot can strengthen the individual's ability to perform certain tasks, reduce the risk of injuries to operators, lower labor costs, and save time, it is widely used in the medical, military, industrial, agricultural, and other fields (Kim *et al.*, 2019). For example, in the medical field, there are many disabled people and elderly people with mobility problems, and they need equipment to help them overcome their physical obstacles and regain their self-care ability. In the industrial field, wearable assistive robots can significantly enhance workers' productivity while simultaneously ensuring their safety on the job.

At the beginning of the 21st century, the U.S. military researched the exoskeleton robot project, dedicated to helping soldiers carry more weapons and equipment and improve the level of protection and movement speed (Yeem *et al.*, 2018). With the development and maturity of the military exoskeleton industry, the research of rehabilitation robots has been promoted. The Hybrid Assistive Limb (HAL) whole-body wearable power assist robot system developed by the University of Tsukuba in Japan can help the elderly and patients with lower limb disabilities to carry out normal squatting, walking, and other activities (Sakurai and Sankai, 2009). The HAL is equipped with a range of ground contact force and angle sensors to track various motion states between the wearer and the exoskeleton. In addition, Nanyang Technological University in Singapore has developed a flexible lower limb rehabilitation robot, which can help the wearer bear large loads and walk long distances (Liu and Low, 2004). Encoders are placed at the joints to measure the joint angle signals while walking, and further control the motor to provide power.

The wearable assistive robot is a typical human–robot interaction system, where accurate recognition and judgment of motion intention is the prerequisite for motion control. Researching the time series modeling and health monitoring of the wearable assistive robot is helpful to better judge the motion intention of people through various sensor signals, to carry out more precise motion control, and to assist people in carrying out corresponding operations.

Many scholars have conducted related research on human movement intention recognition based on sensor data. In 2004, Bao *et al.* used five biaxial accelerometers to obtain the acceleration data of subjects performing 20 daily life actions and used a decision tree classifier to achieve an accuracy of 84% in action recognition (Bao and Intille, 2004). Zhu *et al.* used the hidden Markov models to study the data obtained by the wearable inertial sensor and obtained a good gesture recognition effect (Zhu *et al.*, 2008). Based on the coefficients of the autoregressive model extracted from the acceleration signal and the characteristics of the body tilt angle, Khan *et al.* used the linear discriminant analysis method to reduce the feature dimension and obtained a 97.9% correct rate of human activity recognition by using the artificial neural networks (Khan *et al.*, 2010).

In this chapter, time series data of sensors are used to determine human motion intention, and three types of models, statistical model, intelligent model, and deep learning model, are used for corresponding research. Each type of model contains several different algorithms. The human motion recognition algorithms of wearable assistive robots used in this chapter are shown in figure 6.1.

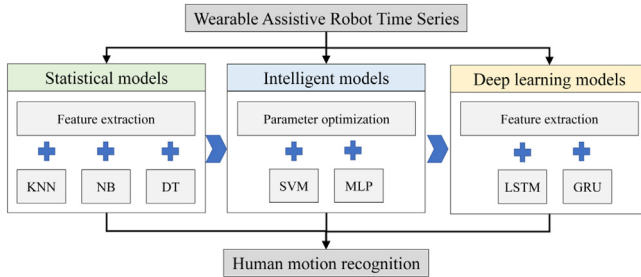


FIG. 6.1 – The human motion recognition algorithms of wearable assistive robots used in chapter 6.

6.2 Wearable Assistive Robot Time Series Measurement

The time series modeling and health monitoring of the wearable assistive robot requires the use of sensor systems to measure related time series signals (Moreno *et al.*, 2008). At present, the motion feedback signals of wearable assistive robots that are often collected can be roughly divided into three categories: attitude signals, force signals, and bioelectric signals. The classification of motion feedback signals of wearable assistive robots is shown in figure 6.2.

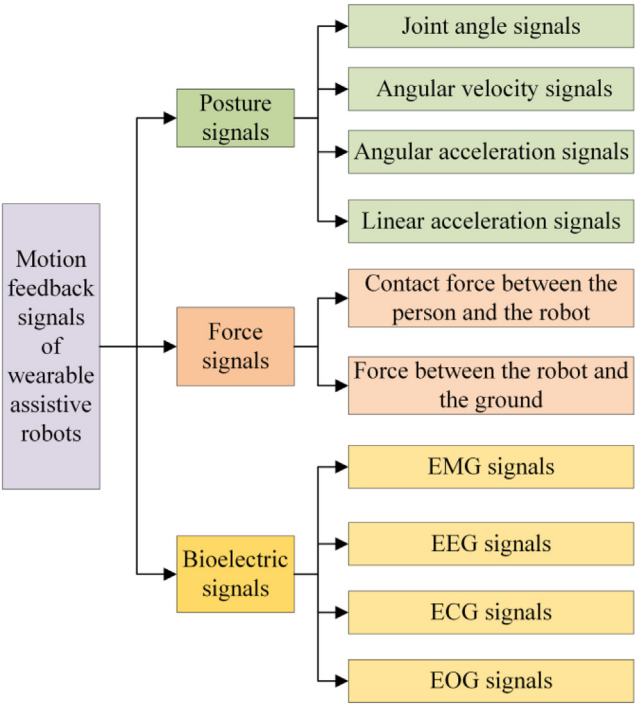


FIG. 6.2 – Classification of motion feedback signals of wearable assistive robots.

In the wearable assistive robot, the attitude signals include joint angle signals, angular velocity signals, angular acceleration signals, linear acceleration signals, etc. The corresponding sensors include absolute encoders and attitude sensors. Among them, the absolute encoder is used in conjunction with the joint motor and is generally installed at the hip and knee joints. It can sense the rotation angle of the motor, thereby detecting the joint angle of the wearable assistive robot, and judging the current motion mode. The attitude sensor mainly integrates a gyroscope and an acceleration sensor. It is usually installed on the waist, thigh, or calf (Vikas and Crane III, 2013). It can measure the angular acceleration and angular velocity signals to obtain the wearer’s motion state, and then determine its motion intention. However, the time series data collected by the attitude sensor has large noise and relies on the filtering algorithm to smooth the waveform, which usually loses the important features of the original data.

When a person wears the wearable assistive robot to perform various movements, there are a variety of forces, including the contact force between the person and the wearable assistive robot, and the force between the robot and the ground. In terms of contact force, the two-dimensional force sensor can be placed on the contact surface of the robot and the human body to measure, and the wearer’s movement intention can be obtained by analyzing the change in the size and distribution of the contact force. However, there are many contact points between the human and the

wearable assistive robot, and it is difficult to accurately locate the contact force. In terms of the force between the robot and the ground, the pressure signals of different parts of the sole can be obtained through multiple pressure sensors. When the wearer's center of gravity changes, the soles of the feet will show different pressure distributions, and the analysis can get the wearer's movement intention (Yang *et al.*, 2019). However, if the wearer is a patient who loses the sense of touch in the lower limbs, he cannot obtain accurate foot pressure information according to the change of his center of gravity, and there is no way to accurately judge the movement intention.

Bioelectric signals include Electromyogram (EMG) signals, Electroencephalogram (EEG) signals, Electrocardiogram (ECG) signals, Electrooculogram (EOG) signals, and so on (Reilly and Lee, 2010). Among them, the research on assistive robots based on EMG and EEG signals is relatively mature. The EMG signals can be collected by the sensing system, which can describe the human body's movement intention. Currently, there are two main ways to collect EMG signals. One way is to insert needle electrodes into the muscles to detect the bioelectric signals, and the other way is to attach electrodes outside the detected muscles to collect the Surface Electromyogram (sEMG). The first method can obtain more accurate results, but it will cause certain trauma to the muscles. The second method is simple and non-invasive (Reaz *et al.*, 2006). The brain is the center of high-level neural activity in the human body. It has massive numbers of neurons connected to transmit and process human body information. By covering the electrode sheet on the outside of the wearer's brain, different EEG signals can be collected when the wearer is thinking about different motion modes (Subasi and Ercelebi, 2005). However, the EEG signal has strong randomness and is very susceptible to interference from the external environment. Therefore, it is difficult to properly process and analyze them.

The sEMG signal acquisition is convenient among the motion feedback signals often collected by wearable assistive robots. In addition, compared with physical sensing signals such as acceleration and pressure, the sEMG signals have strong global information and no measurement lag, which can better and timely reflect the wearer's motion intention. Therefore, the research in this chapter will be carried out based on the sEMG signals detected on the skin surface.

6.3 Wearable Assistive Robot Time Series Uncertainty Analysis

In the process of the application of wearable assistive robots, various uncertain information analysis and processing problems will be faced. Generally speaking, uncertain factors will appear at all levels of the robot system, such as the parameter uncertainty of the robot, the error of the sensor, and external interference.

There may be uncertainties in the parameters of the wearable assistive robot itself. For example, the physical parameters such as the length and mass of the robot connecting rod are unknown, and the actuator of the robot is unreliable due to factors such as mechanical wear and control noise (Wu and Rao, 2007).

The use of sensors to measure the time series data of the wearable assistive robot is the basis for time series modeling. However, affected by physical factors such as their resolution and range, as well as observation noise, sensors will unavoidably interfere in the sensing process, resulting in errors between the obtained observation data and the real situation (Grant, 1991). If the sensor fails, it is difficult to detect the abnormal sensor, and its failure will cause abnormal measurement results.

Wearable assistive robots can be used in various places such as factories, hospitals, roads, etc. Therefore, the environment they face is dynamic, complex, and unpredictable, and may be interfered with by various external conditions (Chen *et al.*, 2011). In a structured environment such as a factory, the environment is low dynamic and its degree of uncertainty is relatively low, but the electronic equipment of the wearable assistive robot is also affected by environmental conditions such as temperature and humidity. In highly dynamic environments such as roads, various vehicles, and pedestrians are constantly changing, and there is no fixed pattern. Excessive external interference may cause the robot to have difficulty computing or even crash.

All of the above uncertain factors will cause the time series information we obtain of the wearable assistive robot to be inaccurate and inadequate, which will affect its time series modeling, health monitoring, and motion control (Wang *et al.*, 2019). Therefore, the emergence of uncertain factors should be minimized. For data problems caused by unavoidable uncertain factors, filtering algorithms can be used to filter out noise from the original data, and outlier detection and processing algorithms can be used to deal with outliers in the original data.

6.4 Wearable Assistive Robot Time Series Statistical Forecasting Method

Human motion recognition based on time-series signals measured by wearable robot sensors can be reduced to a multi-class classification problem. Classification generally analyzes the known training set through the classification algorithm, generates a prediction model, and assigns a category to the unknown category sample in the testing set. In this section, three statistical forecasting algorithms, K-Nearest Neighbor (KNN), Naive Bayes (NB), and Decision Tree (DT), are mainly used for human movement recognition.

6.4.1 Experimental Design

(1) Data description

The sEMG signals used in this chapter are obtained from 8 sensors attached to the surface of the subject's right arm. The four hand motions measured are rock, paper, scissors, and OK, and the first three of them are the same as the gestures with the same name in the game. Each gesture has a total recording time of the 40 s, and the data frequency is 200 Hz, that is, the time interval of each sample point is 5 ms. For each sensor, there are 8000 sample points for each gesture. In this section, 75% of

all sampled data are randomly selected as the training set and the remaining 25% as the testing set. Taking sensor 1 as an example, the data sets of the four gestures are shown in figure 6.3. Taking the stone gesture as an example, the data set of 8 sensors is shown in figure 6.4.

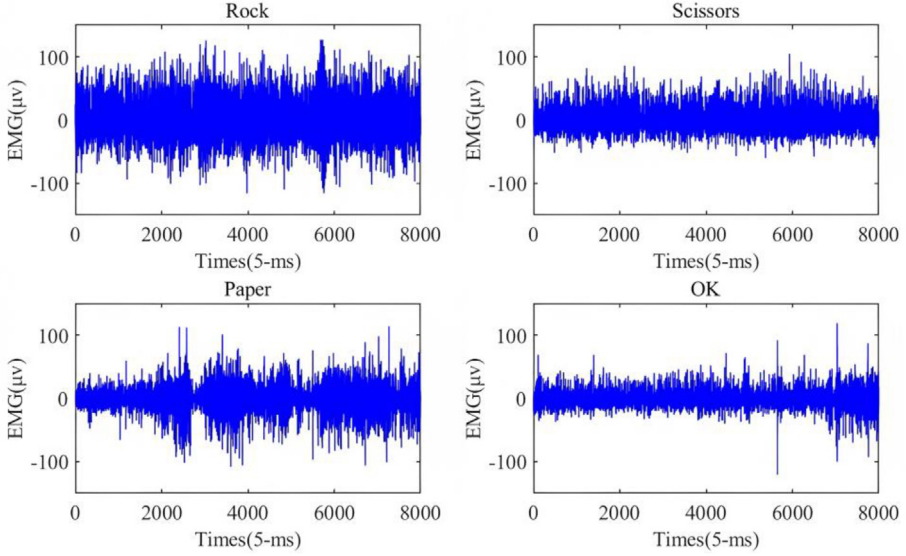


FIG. 6.3 – Four gesture data sets of sensor 1.

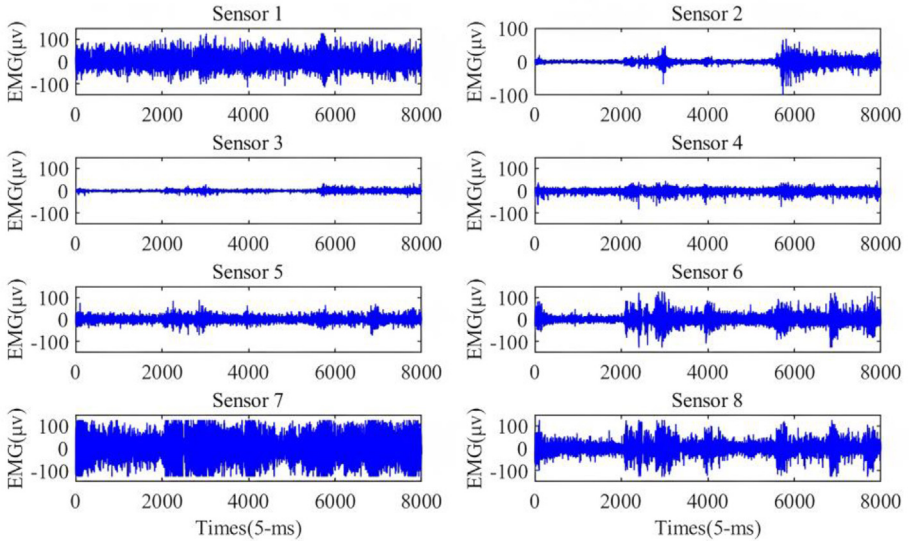


FIG. 6.4 – The stone gesture data set of 8 sensors.

(2) *Evaluation indicator*

In multi-classification problems, accuracy is the most common evaluation indicator, which is the percentage of correctly classified samples in total samples. The calculation of accuracy is shown in equation (6.1). The higher the accuracy, the better the classification effect of the classifier (Baldi *et al.*, 2000).

$$\text{Accuracy} = \frac{1}{n} \sum_{i=1}^n 1(\hat{y}_i = y_i) \quad (6.1)$$

where n is the number of total samples, $\hat{y}_i$ is the forecasting category of the i -th sample, and y_i is the actual category of the i -th sample.

(3) *K-nearest neighbor*

The KNN algorithm can be used to solve both classification and regression problems. By measuring the distance between different eigenvalues, KNN can achieve classification (Peterson, 2009). The classification flow of the KNN algorithm is described in figure 6.5.

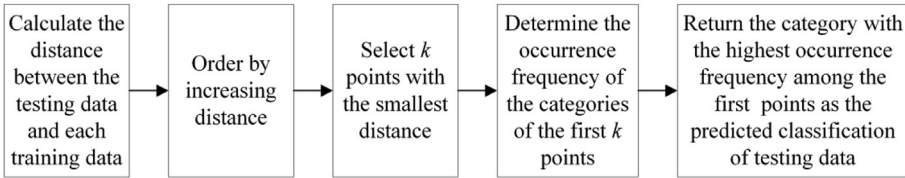


FIG. 6.5 – The classification flow of the KNN algorithm.

In the KNN algorithm, Euclidean distance or Manhattan distance is generally used to calculate the distance between x and y , and the specific calculation equation is shown in equations (6.2) and (6.3) respectively (Guo *et al.*, 2003).

$$d_{\text{euc}}(x, y) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2} \quad (6.2)$$

$$d_{\text{man}}(x, y) = \sum_{k=1}^n |x_k - y_k| \quad (6.3)$$

where x_k and y_k are the k -th eigenvalues of x and y , respectively, and n is the number of the eigenvalue of x or y .

The KNN algorithm is relatively simple, and has its advantages but also disadvantages. The advantages are high accuracy, high tolerance for outliers, and noise. The disadvantages are high computational complexity and high memory requirements.

Its algorithm parameter is k , and the parameter selection needs to be determined according to the data. The larger the value of k is, the greater the deviation of the model will be, and the less sensitive the model will be to noise data. When the value

of k is very large, underfitting may occur. The smaller the value of k is, the larger the variance of the model will be. When the value of k is too small, overfitting will result (Kramer, 2013).

(4) Naive Bayes

The NB algorithm, simplified based on the Bayesian algorithm, is widely used for classification. And the Bayesian algorithm is based on Bayes' theorem. The NB algorithm assumes the independence of the characteristic conditions, which can simplify the calculation of the Bayesian algorithm (Berrar, 2018).

The classification process using the NB algorithm is as follows (Rish, 2001):

1) Assume that the sample to be classified is $x = \{a_1, a_2, \dots, a_n\}$, where n is the feature dimension, that is, the sample to be classified can be divided into n features.

2) There are m different categories in the sample data, and the category set can be expressed as $C = \{y_1, y_2, \dots, y_m\}$.

3) Calculate the prior probability $P(y_i) = N_i/N$ according to the sample data of known classification in the training set, namely the probability of each category, where N_i is the number of samples belonging to the category y_i in the training set, and N is the total number of samples in the training set.

4) Calculate the conditional probability $P(x|y_i)$. Due to the naive Bayes assumption, the following can be obtained:

$$P(x|y_i) = P(a_1, a_2, \dots, a_n|y_i) = P(a_1|y_i)P(a_2|y_i)\dots P(a_n|y_i) = \prod_{j=1}^n P(a_j|y_i) \quad (6.4)$$

5) According to prior probability and conditional probability, the joint probability distribution is obtained, and then the probability value $P(y_i|x)$ of sample x to be classified belongs to each category y_i is calculated. By comparing these probability values, the category with the highest probability value is the category of sample to be classified. The calculation of $P(y_i|x)$ is as follows (Murphy, 2006):

$$P(y_i|x) = \frac{P(x|y_i)P(y_i)}{P(x)} = \frac{\prod_{j=1}^n P(a_j|y_i)P(y_i)}{P(x)} \quad (6.5)$$

6) Since the value of $P(x)$ is fixed, the selected category $\hat{y}$ can be expressed as:

$$\hat{y} = \arg \max_{y_i \in C} \prod_{j=1}^n P(a_j|y_i)P(y_i) \quad (6.6)$$

(5) Decision tree

The DT algorithm is a commonly used classification prediction method in data mining technology. It can summarize decision rules from a series of data with features and labels. The DT is a kind of recognition technology that depends on branch decisions in essence. It is a tree structure that contains a root node, internal node, and leaf node (Song and Ying, 2015).

The index to measure the classification effect of the decision tree is impurity. In general, the lower the impurity, the better the fitting effect of the decision tree on the

training set. Each node in the decision tree has its impurity value, and the impurity of the child node is lower than that of the parent node. Therefore, the impurity of the leaf node is the lowest in the whole decision tree (Safavian and Landgrebe, 1991). There are two commonly used methods to calculate the impurity of sample sets, which can be used to select partition attributes by calculating information entropy or selecting the Gini coefficient. These two methods correspond to two kinds of algorithms for constructing decision trees. Information entropy corresponds to the ID3 algorithm and C4.5 algorithm, and the Gini coefficient corresponds to the Classification and Regression Tree (CART) algorithm.

This section adopts the CART algorithm, which adopts the minimum Gini coefficient to select the split attributes of internal nodes. According to whether the value of the feature is discrete value or continuous value, the DT generated by the CART algorithm can be correspondingly divided into a classification tree and regression tree. Since the CART algorithm is used in the research of the classification problem in this section, the classification tree is adopted.

The steps to form the classification tree in this section are as follows (Breiman *et al.*, 2017):

1) First, set the hyperparameters of the DT algorithm.

2) Calculate the Gini coefficient of each feature in the feature set. Assuming that the entire sample set is S and the feature set is $\{F_1, F_2, \dots, F_n\}$, there are a total of n types of features, and each feature corresponds to a sample set $S_i (1 \leq i \leq n)$. Let j be the number of samples in the sample set S , and k be the number of samples belonging to the feature F_i in the sample set S , then the Gini coefficient calculation of the sample set is as follows:

$$\text{Gini}(S) = \sum_{i=1}^n \sum_{i' \neq i} p_i p_{i'} = 1 - \sum_{i=1}^n p_i^2 \quad (6.7)$$

where $p_i = k/j$ is the probability that a sample in the sample set belongs to feature F_i .

3) Select the feature with the smallest Gini coefficient as the split feature of the root node, and at the same time use the smallest Gini coefficient as the Gini coefficient of the discrete feature, namely:

$$F_* = \arg \min_{F_i \in F} \text{Gini}(S, F_i) \quad (6.8)$$

4) The sample subsets corresponding to the split feature of the root node are used to recursively use the same method as step 2 and step 3 to establish the child nodes of the tree and repeat the cycle until the stopping condition is met. The stopping condition of the recursive operation is that the samples in all sub-nodes belong to the same category, or no feature can be used to divide the node samples or until the threshold set by a certain parameter is reached.

5) Generate the CART classification tree.

The computational complexity of the DT is not high, the calculation amount of the model is small, the calculation speed is faster, the output result is easy to interpret and understand, and the method is not sensitive to missing data.

However, the DT algorithm is prone to overfitting due to too many nodes, and the risk of overfitting is generally reduced by pruning.

6.4.2 Modeling Step

(1) Structure of the statistical models

In this section, three classifiers of KNN, NB, and DT are used to recognize human motions. Firstly, window segmentation is performed on sensor data, and relevant features are extracted from the data in each window to form feature vectors. After normalizing the feature vectors, they can be input into the classifier for training and classification. The specific identification process is shown in figure 6.6.

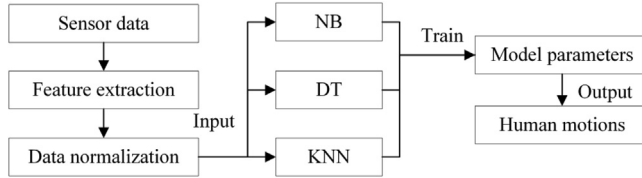


FIG. 6.6 – The classification process of the statistical models.

(2) Data preprocessing

Before performing model training, the sEMG signal measured by each sensor needs to be divided into small time windows. This chapter uses sliding window segmentation technology to divide, and the length of the window is set to 40 ms, that is, each window contains 8 time series sample points. To avoid information loss, this chapter directly uses the vector corresponding to each total time window after the 8 sensor data is merged side by side according to time as a feature vector, and each feature vector has 64 dimensions. An example of the divided time window of the four gestures is shown in figure 6.7.

To improve the training speed and effect, the feature vector is normalized by equation (6.9), and the data is transformed into the interval [0,1].

$$X'_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}} \quad (6.9)$$

(3) Train and forecast

After data preprocessing, the feature vectors and labels are used to train the classifiers KNN, NB, and DT. The parameter settings of these classifiers are shown in table 6.1.

In this section, the learning curves are drawn to determine whether the model used has a bias or variance problem. The learning curve is a graph drawn with the training score and the cross-validation score as a function of the number of training

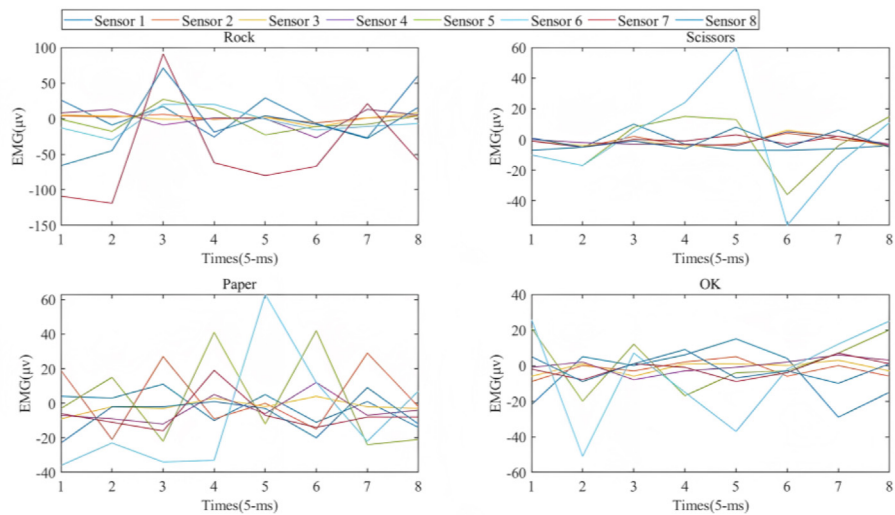


FIG. 6.7 – An example of the divided time window of the four gestures.

TAB. 6.1 – Parameter settings of KNN, NB, and DT.

Classifiers	Parameters	Values
KNN	n_neighbors	10
	algorithm	auto
	leaf_size	10
NB	priors	$P(y_i) = N_i/N$
	var_smoothing	1e-9
DT	criterion	gini
	splitter	best
	max_depth	None

examples. Through cross-validation, limited data is effectively used. The learning curves of KNN, NB, and DT are shown in figures 6.8–6.10.

It can be known from figure 6.8 that the accuracy of the training set and the validation set are not much different from the KNN model, but they are both relatively poor. This shows that the KNN has a high bias problem. In this case, the model is likely to be underfitting, which means that the model cannot fully express the relationship of the data.

As can be seen from figure 6.9, when the NB model is used for prediction, the curves on the training set and the validation set can converge, and the accuracy rate is very high, which shows that NB does not have bias or variance problems and can make accurate predictions for both known and unknown data.

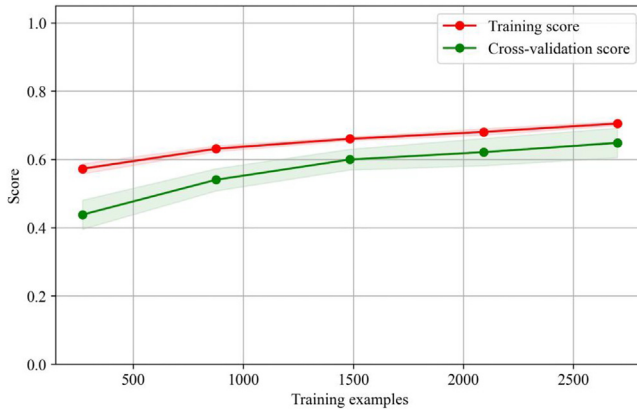


FIG. 6.8 – The learning curves of KNN.

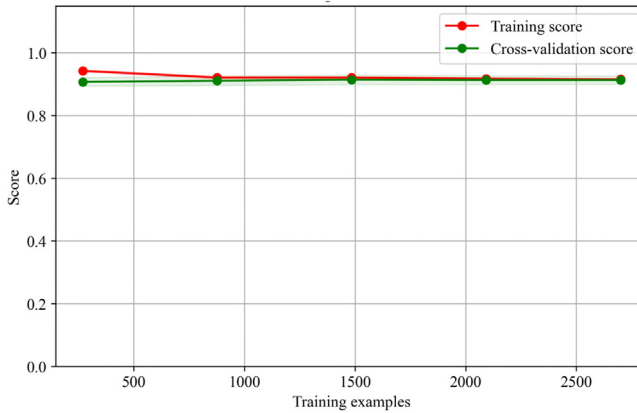


FIG. 6.9 – The learning curves of NB.

It can be known from figure 6.10 that the accuracy of the training set is very high from the DT model, but it differs greatly from the accuracy of the verification set. This indicates that the model has too high variance and can fit the known data well, but the generalization ability is poor, which is likely to be overfitting.

After the KNN, NB, and DT are trained, they are used to forecast the testing set, and the forecasting results are shown in the next section.

6.4.3 Forecasting Results

In this section, the confusion matrix and normalized confusion matrix are used to present the forecasting results. The confusion matrix is a situation analysis table that summarizes the forecasting results of the classification model. The value on the

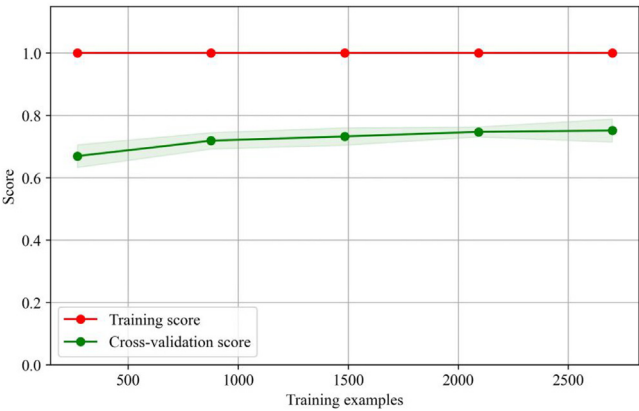


FIG. 6.10 – The learning curves of DT.

diagonal indicates the number of predicted labels equal to the true labels. Therefore, it is very convenient to see the predicted confusion from the confusion matrix, that is, the situation where one category is predicted to become another category. The normalized confusion matrix can get the probability value, and the sum of all the probabilities in each row is 1, so the value of the diagonal represents the recall rate of each category, which can show the effect of prediction more intuitively.

The confusion matrices and normalized confusion matrices of KNN, NB, and DT are shown in figures 6.11–6.13.

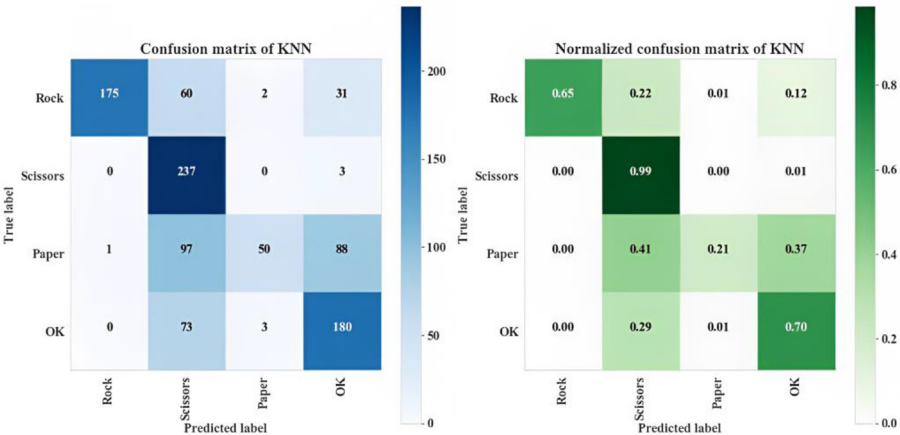


FIG. 6.11 – The confusion matrix and normalized confusion matrix of KNN.

It can be known from figure 6.11 that KNN can achieve good forecasting accuracy for scissors, with a recall rate of 0.99. The recognition effect of rock and OK is barely good, with recall rates are 0.65 and 0.70. However, the accuracy of the paper is relatively poor, the recall rate of which is only 0.21. And the paper is mostly misidentified as scissors and OK.

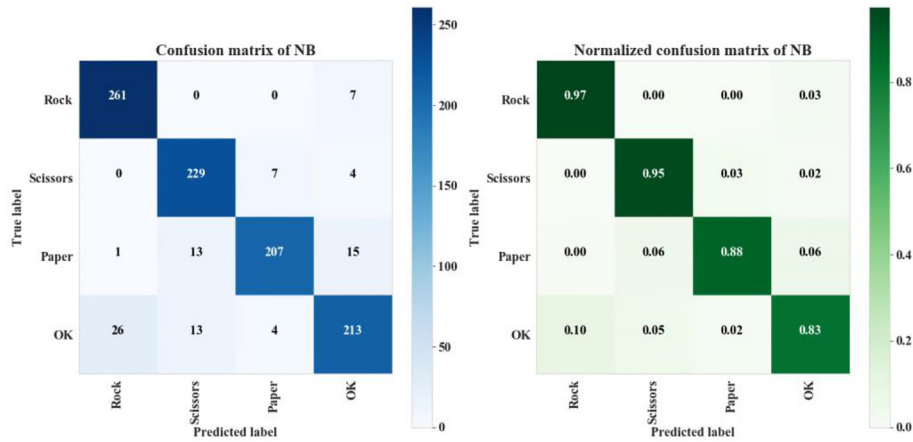


FIG. 6.12 – The confusion matrix and normalized confusion matrix of NB.

It can be known from figure 6.12 that NB can achieve very good forecasting accuracy for the four gestures, and the recall rates of rock, scissors, paper, and OK are 0.97, 0.95, 0.88, and 0.83, respectively.

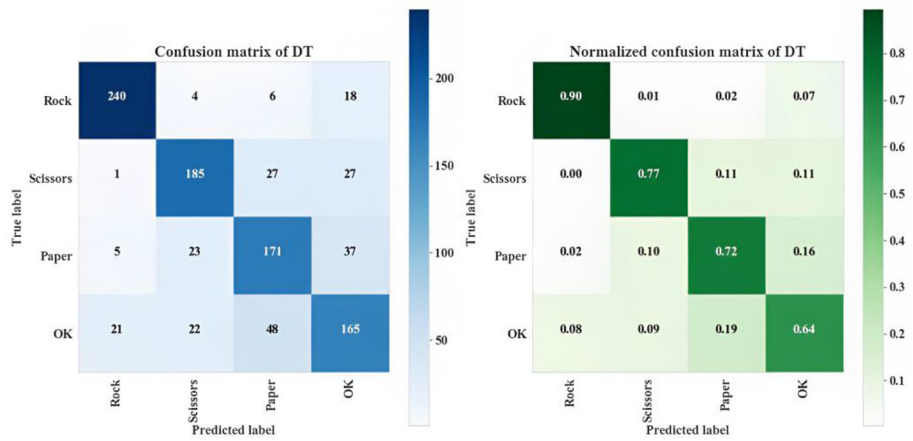


FIG. 6.13 – The confusion matrix and normalized confusion matrix of DT.

It can be known from figure 6.13 that DT has a good recognition effect on all four gestures, and the recall rate of rock, scissors, paper, and OK are 0.90, 0.77, 0.72, and 0.64, respectively.

The classification accuracy and time of the KNN, NB, and DT are shown in table 6.2. The time refers to the total time cost of training and prediction.

TAB. 6.2 – Classification accuracy and time of KNN, NB, and DT.

Classifiers	Accuracy (%)	Time (s)
KNN	64.2	0.753
NB	91.0	0.008
DT	76.1	0.147

From figures 6.11–6.13 and table 6.2, it can be concluded that among the three statistical models, the NB has the best identification performance for the four gestures. The classification accuracy of the NB is quite high, with a value as high as 91.0%, and the time taken by the NB is very short, only 0.008 s. The classification effect of the DT and KNN is not bad, and the DT is a little better than KNN in terms of both prediction accuracy and time cost.

6.5 Wearable Assistive Robot Time Series Intelligent Forecasting Method

6.5.1 Experimental Design

With the rise of machine learning, various intelligent models have been widely used in many fields, and they have a strong nonlinear fitting ability and can achieve good results in classification. In this section, two intelligent forecasting methods, Support Vector Machine (SVM) and Multi-Layer Perceptron (MLP), are mainly used for human movement recognition.

(1) Data description

To make the comparison in section 6.7 more reasonable, this section uses the same data as in section 6.4, with specific data descriptions as described in section 6.4.1.

(2) Support vector machine

The SVM is a learning mechanism that can handle regression and classification problems (Vapnik, 1999). The SVM algorithm was originally used to solve the two-class classification problem, so a suitable multi-classifier is constructed in this section. Direct and indirect methods are commonly used to construct SVM multi-classifiers. Because the direct method is computationally complex and difficult to implement, the indirect method is used in this section. The indirect method is to realize the construction of multiple classifiers by combining multiple binary classifiers, including one-against-one, one-against-all, and other methods (Krebel, 1999).

When using the one-against-one method, only two types of sample data are required as training samples for each classifier, while the one-against-all method requires training all samples for each classifier, so the one-against-one method seems to train a single model faster (Weston and Watkins, 1998). However, the number of binary classifiers required to be constructed and tested by the one-against-one method increases as a quadratic function concerning the number of categories m , and the total training time and testing time are relatively long.

There are only four categories in this section, so the one-against-one method is adopted, that is, a binary SVM classifier is constructed between every two types of data (Knerr *et al.*, 1990). When the number of categories is m , this method will construct $m(m-1)/2$ classifiers. When classifying an unknown sample, these $m(m-1)/2$ classifiers are used for classification at a time, and the class that gets the most votes, in the end, determines which class the unknown sample belongs to.

When constructing a binary SVM classifier between the i -th type and j -th type samples, the problem can be expressed as (Hsu and Lin, 2002):

$$\begin{aligned} \min \quad & \frac{1}{2} \|w^{ij}\|^2 + C \sum_{n=1}^l \xi_n^{ij} (w^{ij})^T \\ \text{s.t.} \quad & \begin{cases} 1 - (w^{ij})^T \phi(x_n) - b^{ij} \leq \xi_n^{ij}, & \text{if } y_n = i \\ 1 + (w^{ij})^T \phi(x_n) + b^{ij} \leq \xi_n^{ij}, & \text{if } y_n = j \\ \xi_n^{ij} \geq 0 \end{cases} \end{aligned} \quad (6.10)$$

where $\{(x_n, y_n), n = 1, 2, \dots, l\}$ is the training sample, n represents the index of the sample in the union of the i -th and j -th type, w^{ij} is the plane normal vector of the function, C is the regularization parameter, ξ_n^{ij} is the slack variable, b^{ij} is the offset of the functioning plane, and ϕ represents the nonlinear mapping from the input space to the feature space.

(3) Multilayer perceptron

The MLP is a feed-forward artificial neural network, developed from perceptron (Pal and Mitra, 1992). In the construction of the MLP model, the weights w and the bias b are randomly initialized in advance, and then the perceptron is learned. The learning process is to continuously improve w and b in the direction of reducing errors, that is, to minimize the loss of function value. Its mathematical expression is (Bishop, 1995):

$$y_i = f\left(\sum_{j=1}^n w_{ij} \times x_j + b_j\right), \quad i = 1, 2, \dots, m, \quad j = 1, 2, \dots, n \quad (6.11)$$

where f represents the activation function, y_i represents the output of the i -th neuron in the output layer, w_{ij} represents the connection weight between the j -th input neuron and the i -th output neuron, x_j represents the input feature signal, and b_i represents the bias of the i -th neuron in the output layer.

The MLP has high computational efficiency, strong classification capabilities, and advantages in non-linear mapping, so it is widely used in fields such as data classification, data prediction, and function estimation (Murtagh, 1991).

6.5.2 Modeling Step

(1) Structure of the intelligent models

In this section, two classifiers of SVM and MLP are used to recognize human motions. As in section 6.4.2, window segmentation is used on original data, and feature extraction is used to get feature vectors. After data normalization of the feature vectors, the classifiers are trained by them. To optimize the performance of the model, this section adjusts the parameters through the grid search method and obtains the best-performing model parameters through loop traversal among all optional parameter combinations. Finally, the trained intelligent model is used for prediction. The structure of the intelligent models is shown in figure 6.14.

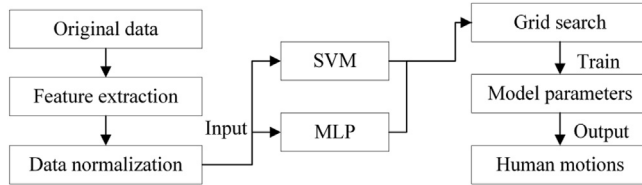


FIG. 6.14 – The classification process of the intelligent models.

(2) Data preprocessing

The data preprocessing process in this section is the same as in section 6.4.2.

(3) Train and forecast

After data preprocessing, the feature vectors and labels are used to train the classifiers SVM and MLP. The parameter settings are shown in table 6.3.

It can be seen from table 6.3 that three SVM models and MLP models are constructed in this section, named SVM1, SVM2, SVM3, MLP1, MLP2, and MLP3, and their parameters are different. In the three SVM models, the parameters of SVM1 are arbitrarily set, while the parameters of SVM2 and SVM3 are obtained through grid search. The kernel function of SVM2 is restricted to the radial basis kernel function, and the regularization parameter C belongs to [0.01,0.1,1,10]. The kernel function of SVM3 is restricted to the polynomial kernel function, the regularization parameter C belongs to [0.01,0.1,1,10], and the degree of polynomial kernel function belongs to range(1,5).

In the three MLP models, the parameters of MLP1 and MLP2 are arbitrarily set, while the parameters of MLP3 are obtained by grid search. The number of neurons in the hidden layer of MLP3 is selected from [(64,),(50,50),(32,64),(64,32),(100,)], the activation function of the hidden layer is selected from ['identity', 'logistic', 'tanh', 'relu'], the solver for weight optimization is selected from ['lbfgs', 'sgd', 'adam'], and the maximum number of iterations is selected from [200,500].

TAB. 6.3 – Parameter settings of SVM and MLP.

Classifiers	Parameters	Values
SVM1	C	1
	kernel	poly
	degree	3
SVM2	C	10
	kernel	rbf
SVM3	C	10
	kernel	poly
	degree	2
MLP1	hidden layer sizes	(100,)
	activation	relu
	solver	adam
	max_iter	200
MLP2	hidden_layer_sizes	(50,50)
	activation	tanh
	solver	adam
	max_iter	500
MLP3	hidden layer sizes	(100,)
	activation	relu
	solver	adam
	max_iter	500

In this section, the learning curves are also used to judge the data-fitting effect of the intelligent models. The learning curves of three kinds of SVM models are shown in figures 6.15–6.17. The learning curves of three kinds of MLP models are shown in figures 6.18–6.20.

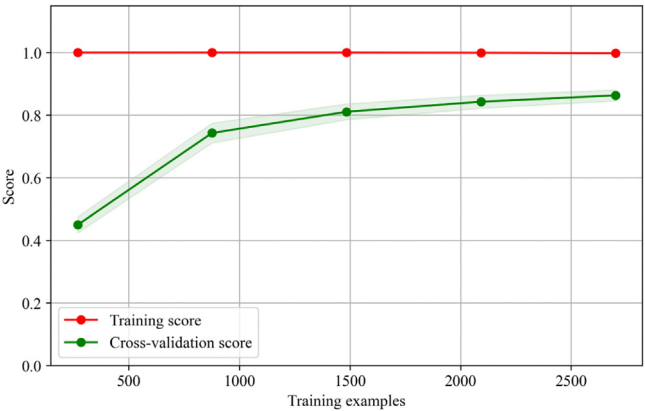


FIG. 6.15 – The learning curves of SVM1.

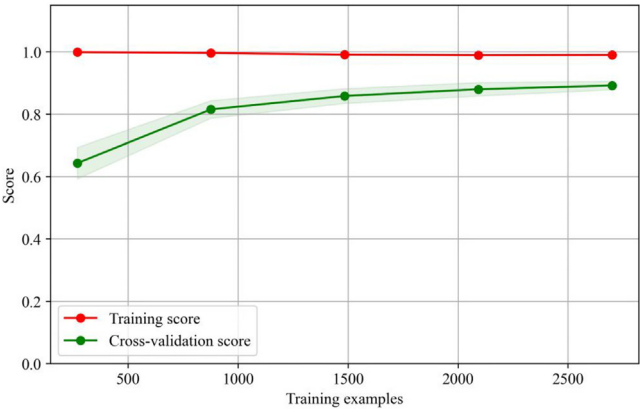


FIG. 6.16 – The learning curves of SVM2.

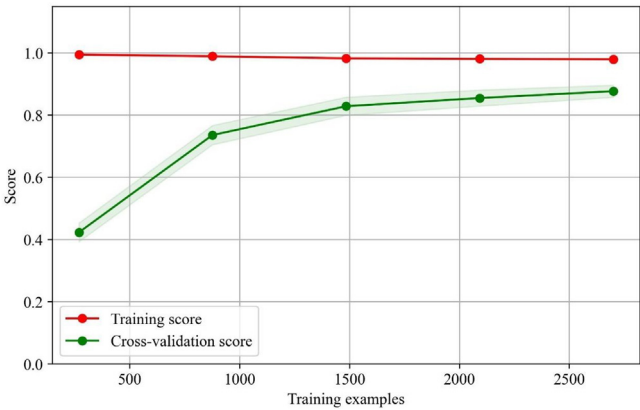


FIG. 6.17 – The learning curves of SVM3.

It can be known from figures 6.15–6.17 that when the three kinds of SVM models are used, with the increasing of training examples, the learning curves can converge. However, the learning curves of SVM1 show that the variance of SVM1 is high. Compared with SVM1, the SVM2 and SVM3 can fit the unknown data better, which indicates that the grid search is useful to find better parameters.

It can be known from figures 6.18–6.20 that when three kinds of MLP models are used, both the training score and cross-validation score go up as the training examples increase, and the scores of MLP1 and MLP3 still have a growing trend. The learning curves of MLP3 converge and the scores of MLP3 are relatively high, which shows the effect of grid search.

After the SVM and MLP are trained, they are used to forecast the testing set, and the forecasting results are shown in the next section.

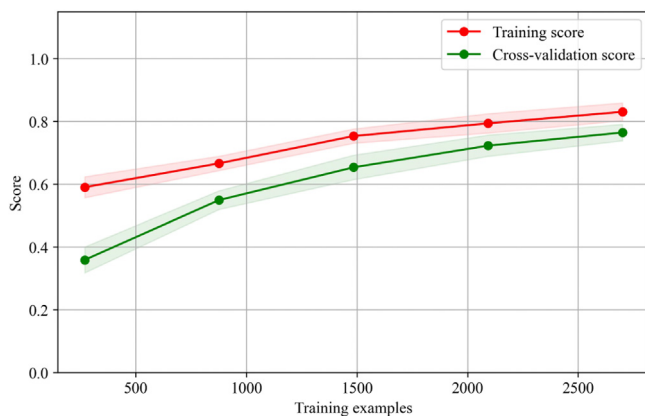


FIG. 6.18 – The learning curves of MLP1.

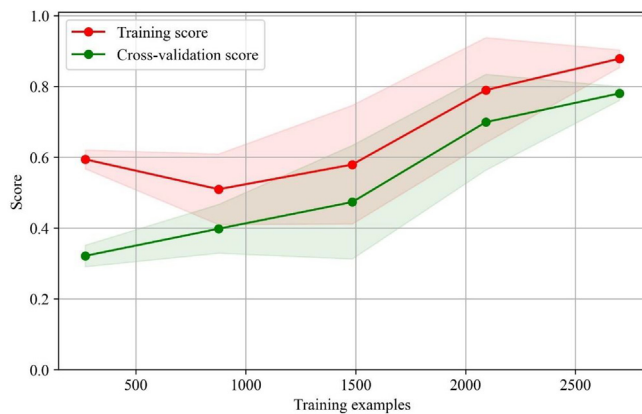


FIG. 6.19 – The learning curves of MLP2.

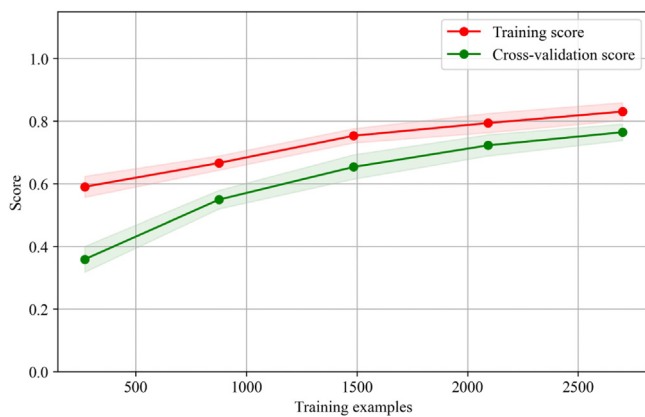


FIG. 6.20 – The learning curves of MLP3.

6.5.3 Forecasting Results

In this section, the confusion matrix and normalized confusion matrix are also used to present the forecasting results of intelligent models, which are shown in figures 6.21–6.23 of the SVM models and figures 6.24–6.26 of the MLP models respectively.

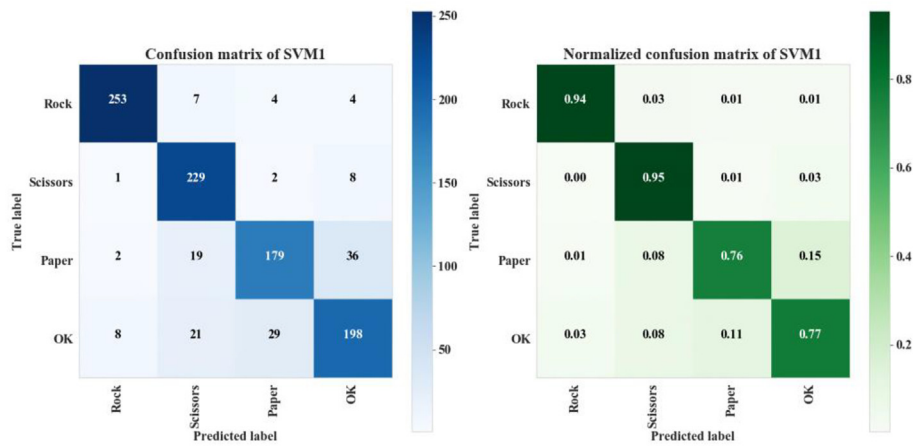


FIG. 6.21 – The confusion matrix and normalized confusion matrix of SVM1.

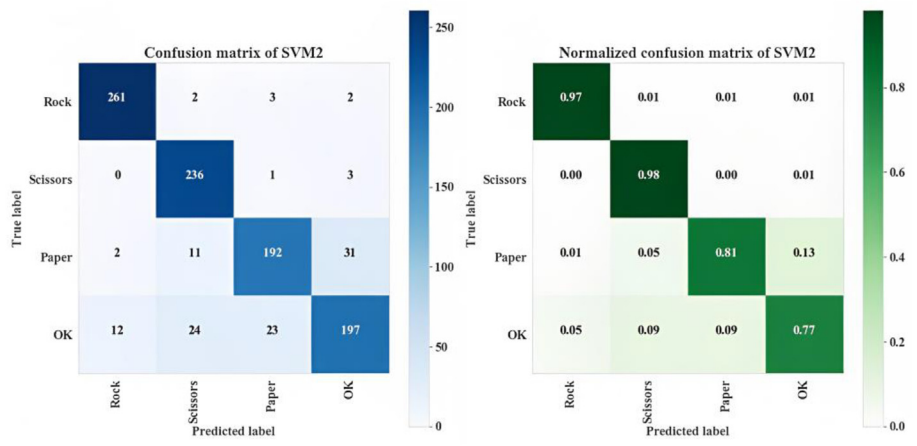


FIG. 6.22 – The confusion matrix and normalized confusion matrix of SVM2.

It can be known from figures 6.21–6.23 that all three SVM models can achieve very good classification results for rock and scissors, with a recall rate greater than 0.93. In addition, the forecasting effect of paper and OK is good, and their recall rates are all around 0.8.

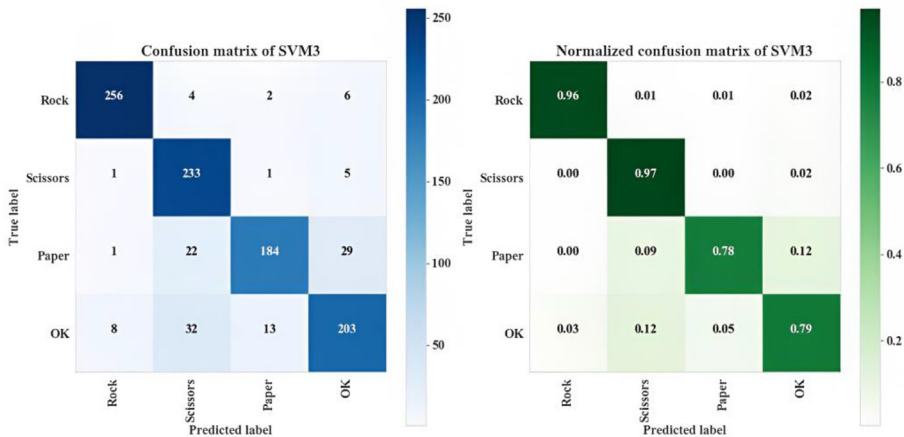


FIG. 6.23 – The confusion matrix and normalized confusion matrix of SVM3.

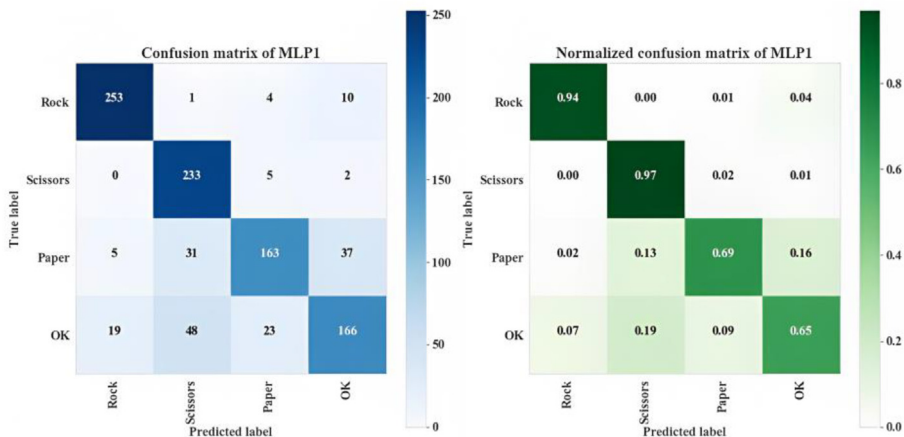


FIG. 6.24 – The confusion matrix and normalized confusion matrix of MLP1.

It can be known from figures 6.24–6.26 that all three MLP models can achieve very good forecasting results for rock and scissors, and their recall rates are all greater than 0.88. In addition, the recognition effect of paper and OK is not bad, and the recall rate is 0.64 at the lowest and 0.86 at the highest.

The classification accuracy and time of SVM and MLP are shown in table 6.4.

From figures 6.21–6.26 and table 6.4, it can be known that all of the six models can achieve good forecasting results, and their classification accuracy is above 80%. Among the three SVM models, SVM2 and SVM3 with grid search for parameter adjustment can achieve better results than SVM1 with arbitrary parameters, and the SVM2 with the radial basis kernel function can obtain higher classification

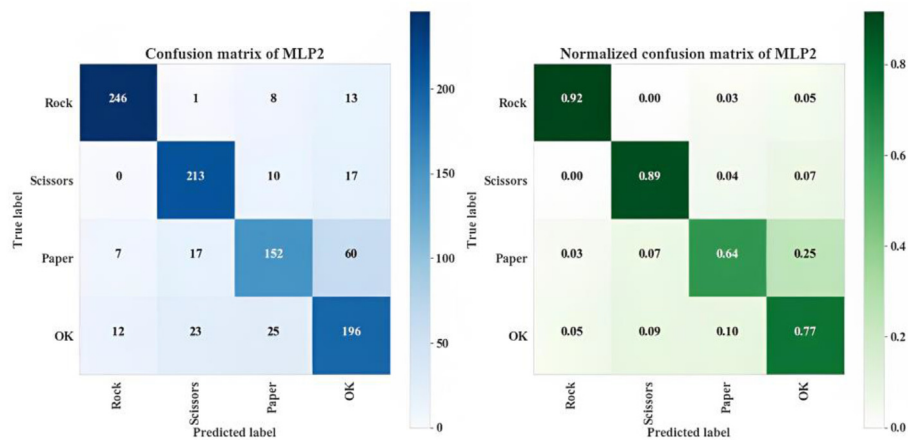


FIG. 6.25 – The confusion matrix and normalized confusion matrix of MLP2.

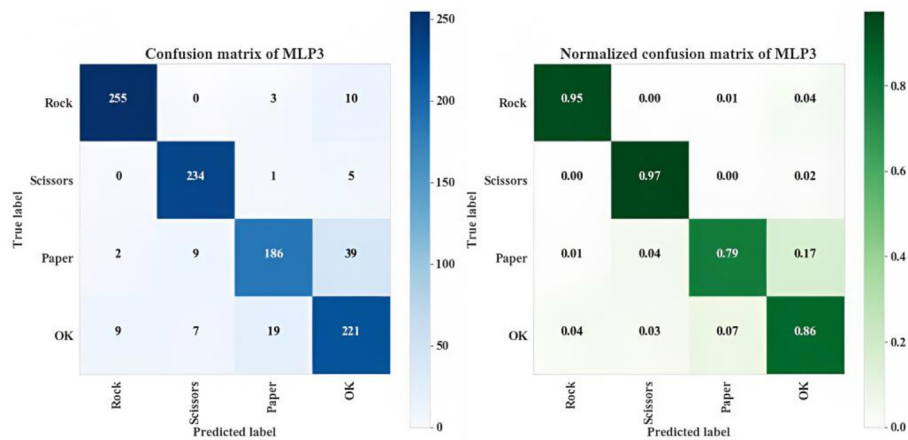


FIG. 6.26 – The confusion matrix and normalized confusion matrix of MLP3.

TAB. 6.4 – Classification accuracy and time of SVM and MLP.

Classifiers	Accuracy (%)	Time (s)
SVM1	85.9	2.63
SVM2	88.6	10.1
SVM3	87.6	58.9
MLP1	81.5	2.98
MLP2	80.7	7.52
MLP3	89.6	991

accuracy than the SVM3 with the polynomial kernel function. Among the three MLP models, MLP3 with grid search for parameter adjustment can achieve better results than MLP1 and MLP2, which proves the effect of grid search. MLP3 has the highest accuracy and the longest time cost, 89.6% and 991 s, respectively. This is because the grid search improves the accuracy and also increases the time cost.

6.6 Wearable Assistive Robot Time-Series Deep Learning Forecasting Method

6.6.1 Experimental Design

Deep learning has a deep network structure and can automatically learn the features required by the target task from the input data. This is more advantageous than traditional machine learning methods that rely on prior experience to manually extract features, and greatly reduces the workload. In this section, two deep learning forecasting methods, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), are mainly used for human movement recognition.

(1) Data description

To make the comparison in section 6.7 more reasonable, this section uses the same data as in section 6.4, with specific data descriptions as described in section 6.4.1.

(2) Long short-term memory

The Recurrent Neural Network (RNN) can memorize the previous information and apply it as network input (Pascanu *et al.*, 2013). However, the RNN has the problems of gradient vanishing and explosion (Zhang *et al.*, 2020). It is not effective when dealing with long sequences, and it is difficult to learn the long-term dependence of time series (Hochreiter, 1998). As a variant of RNN, the LSTM retains the advantages of standard RNN and can solve the problem of long-term dependence on the RNN (Qin *et al.*, 2018).

The LSTM uses memory cells to store and output information and has a special structure called a ‘gate’ to remove or add information to the cell and update the cell state of the LSTM according to the activation of the gate (Graves, 2012). The input provided to LSTM will be fed back to three different gates, namely, the input gate, forget gate, and output gate, which controls the reading of new information, discarding information, and output information respectively.

Suppose i_t represents the activation vector of the input gate, x_t is the input of this unit, h_{t-1} is the output of the previous unit, W represents the weight matrix between the gates, b represents the bias of each gate, and σ represents the activation function, the calculation of the input gate is as follows (Hochreiter and Schmidhuber, 1997):

$$i_t = \sigma_i(W_i \bullet [h_{t-1}, x_t] + b_i) \quad (6.12)$$

Suppose f_t represents the activation vector of the forget gate, the calculation of the forget gate is as follows (Hochreiter and Schmidhuber, 1997):

$$f_t = \sigma_f(W_f \bullet [h_{t-1}, x_t] + b_f) \quad (6.13)$$

Assuming that c_t represents the activation vector of the cell, c_{t-1} represents the cell state at the previous moment, the calculation for cell state update is as follows (Hochreiter and Schmidhuber, 1997):

$$c_t = f_t c_{t-1} + i_t \sigma_c(W_c \bullet [h_{t-1}, x_t] + b_c) \quad (6.14)$$

Assuming that o_t represents the activation vector of the output gate, h_t is the output of this unit, then the calculation of the output gate is as follows (Hochreiter and Schmidhuber, 1997):

$$o_t = \sigma_o(W_o \bullet [h_{t-1}, x_t] + b_o) \quad (6.15)$$

$$h_t = o_t \sigma_h(c_t) \quad (6.16)$$

The LSTM improves the long-term dependency problem existing in RNN and is very suitable for dealing with time series problems. However, each cell of LSTM has 4 fully connected layers. The LSTM may consume a lot of time if the network is very deep.

(3) Gated recurrent unit

The GRU is a variant of LSTM, which can solve the long dependency problem in the RNN network, and has a simple structure. The GRU only has two gates, a reset gate and an update gate. The reset gate controls the retention ratio of information at the current moment and retains key recent information. The update gate controls the amount of the previous memory retained in the current state and retains the key information in the long-term memory (Chung *et al.*, 2014).

Suppose r_t represents the activation vector of the reset gate, h_{t-1} is the output of the previous unit, x_t is the current input, W represents the weight matrix, and σ represents the activation function, the calculation of the reset gate is as follows (Zhao *et al.*, 2017):

$$r_t = \sigma_r(W_r \bullet [h_{t-1}, x_t]) \quad (6.17)$$

Suppose z_t represents the activation vector of the update gate, the calculation of the update gate is as follows (Zhao *et al.*, 2017):

$$z_t = \sigma_z(W_z \bullet [h_{t-1}, x_t]) \quad (6.18)$$

Assuming that $\tilde{h}_t$ represents the candidate output of the current unit, h_t is the output of the current unit, then the calculation of output is as follows (Zhao *et al.*, 2017):

$$\tilde{h}_t = \sigma_h(W_h \bullet [r_t * h_{t-1}, x_t]) \quad (6.19)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \quad (6.20)$$

Compared with LSTM, the GRU model has a simpler structure and fewer parameters, which can reduce the risk of overfitting, and the training speed will also be improved.

6.6.2 Modeling Step

(1) Structure of the deep learning models

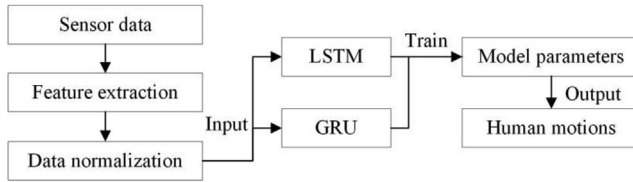


FIG. 6.27 – The classification process of the deep learning models.

In this section, two classifiers of LSTM and GRU are used for classification research. As in section 6.4.2, this section uses the same process to obtain the feature vectors. The deep learning models are trained by the normalized feature vectors. Finally, the forecasting results are obtained by the trained models. The classification process of the deep learning models is shown in figure 6.27.

(2) Data preprocessing

The data preprocessing process in this section is the same as in section 6.4.2.

(3) Train and forecast

After data preprocessing, the feature vectors and labels are used to train the classifiers LSTM and GRU. The parameter settings of them are shown in table 6.5.

In this section, the loss and accuracy curves are used to evaluate the data-fitting effect of the deep learning models. The loss and accuracy curves of LSTM are shown in figures 6.28 and 6.29. The loss and accuracy curves of GRU are shown in figures 6.30–6.31.

It can be known from figure 6.28 that the training and validation loss curves of LSTM are in a downward trend with the increase of epoch. However, the loss curve of the validation data fluctuates, and fluctuates greatly at the beginning and less later. It can be seen from figure 6.29 that the training and validation accuracy curves of LSTM are in an upward trend with the increase of epoch, and there are also fluctuations.

TAB. 6.5 – Parameter settings of LSTM and GRU.

Classifiers	Layers	Parameters	Values
LSTM	/	optimizer	RMSprop
		epochs	100
		batch_size	8
	Layer1	units	64
		input_shape	(8,8)
		dropout	0.2
	Layer 2	units	64
		dropout	0.2
		units(dense)	100
	Layer 3	units	64
		dropout	0.2
		units(dense)	4
		activation(dense)	softmax
GRU	/	optimizer	adam
		epochs	100
		batch_size	8
	Layer 1	units	32
		input_shape	(8,8)
		dropout	0.2
	Layer 2	units	100
		dropout	0.2
	Layer 3	units	64
		dropout	0.2
		units(dense)	4
		activation(dense)	softmax

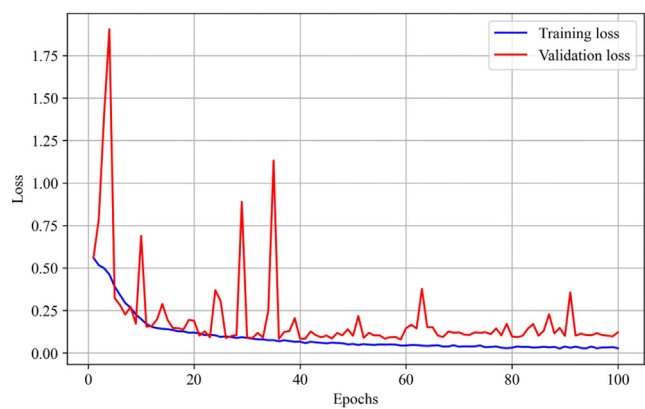


FIG. 6.28 – The loss curves of LSTM.

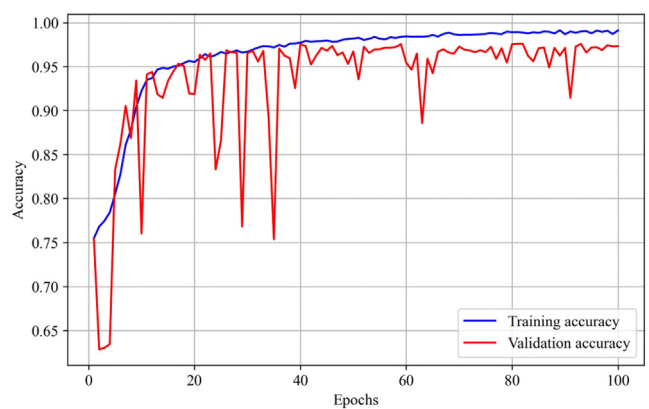


FIG. 6.29 – The accuracy curves of LSTM.

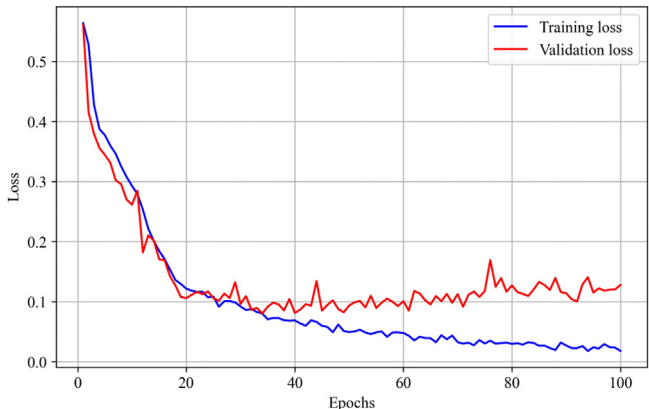


FIG. 6.30 – The loss curves of GRU.

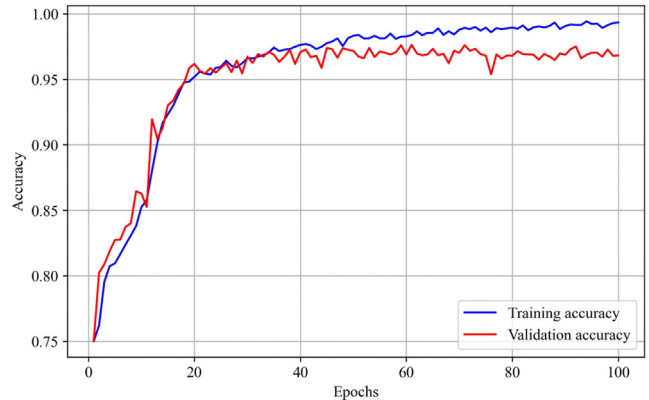


FIG. 6.31 – The accuracy curves of GRU.

As can be seen from figure 6.30, when the GRU model is used for classification, the loss curves of training and validation drop and level off as the epoch increases. As can be seen from figure 6.31, when the GRU model is used for classification, the accuracy curves of training and validation rise and level off as the epoch increases. The curves of GRU also fluctuate, but compared with LSTM, the curves of GRU fluctuate less.

After the LSTM and GRU are trained, they are used to forecast the testing set, and the forecasting results are shown in the next section.

6.6.3 Forecasting Results

In this section, the confusion matrix and normalized confusion matrix are also used to present the forecasting results of intelligent models. The confusion matrices and normalized confusion matrices of LSTM and GRU are shown in figures 6.32 and 6.33.

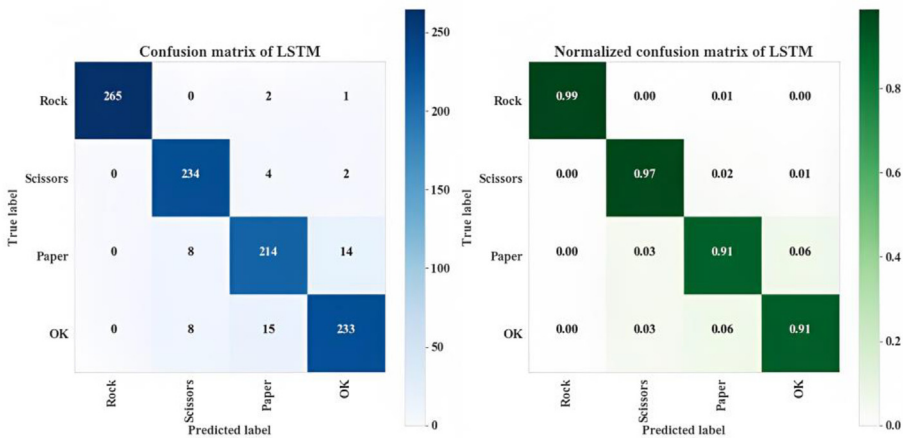


FIG. 6.32 – The confusion matrix and normalized confusion matrix of LSTM.

It can be known from figure 6.32 that LSTM can achieve very good classification accuracy for rock, scissors, paper, and OK, and their recall rates are all greater than 0.90, respectively 0.99, 0.97, 0.91, and 0.91.

It can be known from figure 6.33 that GRU can also achieve very good forecasting accuracy for the four gestures, and the recall rate of rock, scissors, paper, and OK are 1.00, 0.97, 0.92, and 0.84, respectively.

The classification accuracy and time of LSTM and GRU are shown in table 6.6. The time refers to the total time cost of training and prediction.

From figures 6.32, 6.33 and table 6.6, it can be known that LSTM can achieve a very high gesture recognition accuracy of 94.6%, which is higher than the GRU model. However, the classification accuracy of the GRU model is not low,

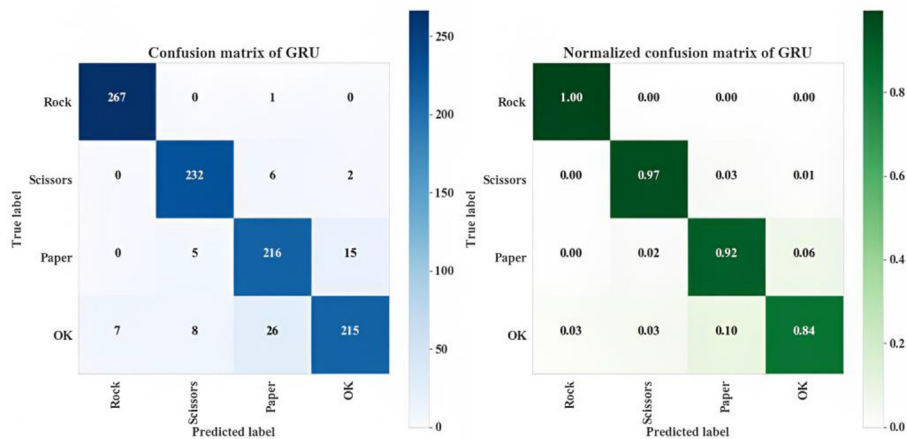


FIG. 6.33 – The confusion matrix and normalized confusion matrix of GRU.

TAB. 6.6 – Classification accuracy and time of LSTM and GRU.

Classifiers	Accuracy (%)	Time (s)
LSTM	94.6	982
GRU	93.0	723

which is 93.0%. And because the structure of GRU is simpler, the total time spent on training and prediction of the GRU model is less than that of LSTM.

6.7 Comparative Analysis of Forecasting Performance

Sections 6.4–6.6 introduce several different forecasting models. The sEMG signals collected by the wearable assistive robot sensor are used for human motion recognition. The human motion recognition algorithms of wearable assistive robots used in this chapter can be divided into three types: statistical forecasting method (KNN, NB, DT), intelligent forecasting method (SVM, MLP), and deep learning forecasting method (LSTM, GRU). The forecasting accuracy and the time cost of different models are shown in figures 6.34 and 6.35. The SVM and MLP with the highest forecasting accuracy among the three models with different parameters are selected as their representative models.

It can be concluded from figures 6.34 and 6.35 that the forecasting effect achieved by the statistical forecasting models varies widely, among which the accuracy of KNN and DT is around 70%, while that of NB is over 90%. But all three statistical models take very little time, less than a second. The two intelligent models used in this chapter have a classification accuracy of close to 90%, which is better than most statistical models, but the time cost is longer because of grid search. Among the seven models used in this chapter, the prediction accuracy of the two deep learning methods

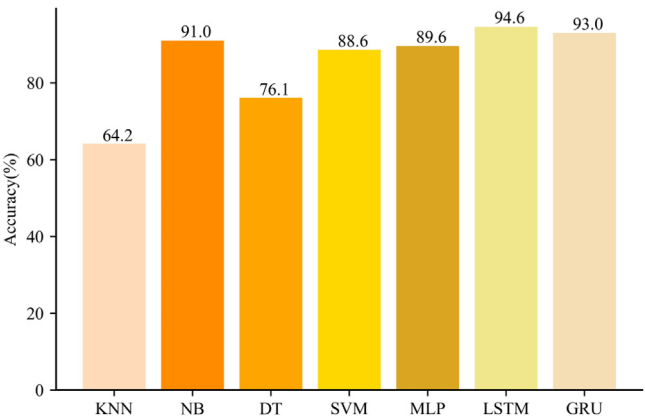


FIG. 6.34 – The forecasting accuracy of different models.

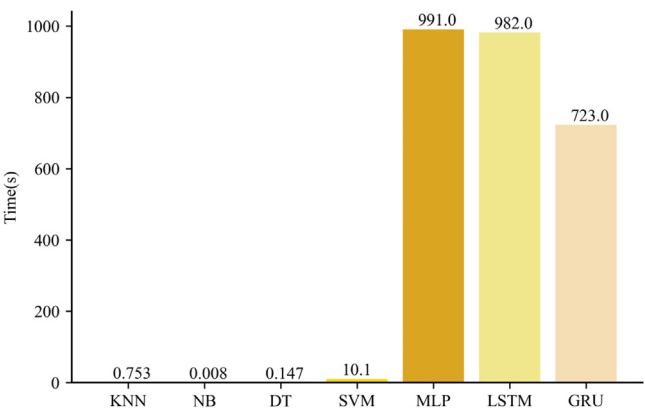


FIG. 6.35 – The time cost of different models.

ranked the top two, the LSTM accuracy is as high as 94.6%, and the time cost is 982 s. This is because the deep learning model has multiple hidden layers, has a strong feature self-learning ability, and can learn more high-level and more meaningful features. Moreover, the deep network structure is more suitable for progressive learning, and the finally obtained network data is more conducive to classification, so a good classification effect can be achieved, and it takes a relatively long time.

6.8 Wearable Assistive Robot Motion Control Based on Forecasting

The wearable assistive robot is different from other independently controlled robots. It is a typical human–robot interaction system. It can not only assist the wearer to complete complex tasks that ordinary industrial robots cannot complete, but also

provide assistance to the human body and enhance the wearer's athletic ability (Anam and Al-Jumaily, 2012). Since the object of the action is a person, it is necessary to predict the person's intention in advance and make corresponding exercises in time to assist. Therefore, accurate forecasting of motion intention is the prerequisite for realizing the motion control of the wearable assistive robot. For example, when a wearable assistive robot is used as a rehabilitation robot, there are generally two rehabilitation training modes, passive training, and active training. Passive training allows the wearable assistive robot to move according to a predetermined trajectory, while active training requires the robot to recognize the wearer's movement intention and then change the movement state to adapt to the change of movement intention (Li *et al.*, 2014).

The reasonable motion control design guarantees to realization of the coordinated motion and the consistent motion between the wearer and the wearable assistive robot. The conventional motion control structure of the wearable assistive robot includes three levels, namely the task level, the interaction level, and the execution level (Proietti *et al.*, 2015). The specific motion control structure of a wearable assistive robot is shown in figure 6.36.

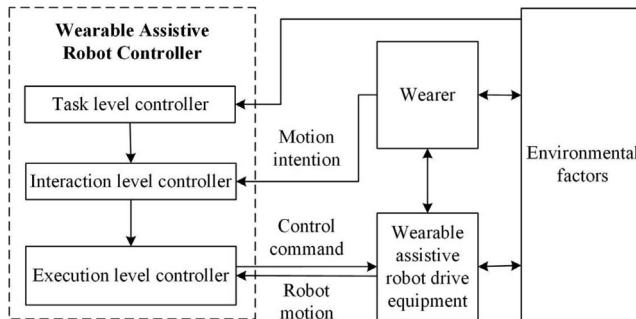


FIG. 6.36 – The motion control structure of the wearable assistive robot.

6.9 Conclusions

The wearable assistive robot is a kind of robot widely used in military, industrial, medical, and other fields. Its application can effectively reduce the risk of injury to workers, reduce time costs, and have good economic benefits. The motion control of the wearable assistive robot requires accurate recognition of the motion intention of the human body. This chapter introduces the time series forecasting methods of wearable assistive robots in detail, including statistical forecasting methods, intelligent forecasting methods, and deep learning forecasting methods to recognize human movement intentions. The following conclusions can be drawn from the experimental results:

- (a) Among the three statistical prediction models, KNN is underfitting and has the lowest prediction accuracy of only 64.2%; DT is overfitting and has a prediction

- accuracy of 76.1%; NB has the highest prediction accuracy, which is 91%, and the time cost is lowest, only 0.008 s.
- (b) In the intelligent model, three models with different parameters are constructed for each of the SVM and MLP in this chapter. The classification accuracy of all six models is higher than 80%, which is better than most statistical models. This chapter uses grid search for parameter optimization, which effectively improves the prediction accuracy, but it also takes more time.
 - (c) The motion recognition effect of the deep learning method used in this chapter is better than all other models in this chapter. In the two deep learning models, LSTM can achieve a higher classification accuracy of 94.6%, while the GRU prediction accuracy is 93%, which is also very high, and the GRU structure is simpler and takes less time than LSTM.

References

- Anam K., Al-Jumaily A. A. (2012) Active exoskeleton control systems: State of the art, *Proc. Eng.* **41**, 988–994.
- Baldi P., Brunak S., Chauvin Y., Andersen C. A., Nielsen H. (2000) Assessing the accuracy of prediction algorithms for classification: An overview, *Bioinformatics* **16**(5), 412–424.
- Bao L., Intille S. S. (2004) *Activity recognition from user-annotated acceleration data*. Paper presented at the International conference on pervasive computing.
- Berrar D. (2018) Bayes' theorem and naive Bayes classifier, *Encyclopedia of bioinformatics and computational biology: ABC of bioinformatics*. Elsevier Science Publisher, Amsterdam, The Netherlands, pp. 403–412.
- Bishop C. M. (1995) *Neural networks for pattern recognition*. Oxford University Press.
- Breiman L., Friedman J. H., Olshen R. A., Stone C. J. (2017) *Classification and regression trees*. Routledge.
- Chen S., Li Y., Kwok N. M. (2011) Active vision in robotic systems: A survey of recent developments, *Int. J. Rob. Res.* **30**(11), 1343–1377.
- Chung J., Gulcehre C., Cho K., Bengio Y. (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- Grant E. (1991) Uncertainty in robot sensing, *Sensor-based robots: Algorithms and architectures*. Springer, pp. 41–57.
- Graves A. (2012) Long short-term memory, *Supervised sequence labelling with recurrent neural networks*. Springer, pp. 37–45.
- Guo G., Wang H., Bell D., Bi Y., Greer K. (2003) *KNN model-based approach in classification*. Paper presented at the OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”.
- Hochreiter S. (1998) The vanishing gradient problem during learning recurrent neural nets and problem solutions, *Int. J. Uncertainty Fuzziness Knowl. Based Syst.* **6**(02), 107–116.
- Hochreiter S., Schmidhuber J. (1997) Long short-term memory, *Neural Comput.* **9**(8), 1735–1780.
- Hsu C.-W., Lin C.-J. (2002) A comparison of methods for multiclass support vector machines, *IEEE Trans. Neural Networks* **13**(2), 415–425.
- Khan A. M., Lee Y.-K., Lee S. Y., Kim T.-S. (2010) A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer, *IEEE Trans. Inf. Technol. Biomed.* **14**(5), 1166–1172.
- Kim H. S., Koo D. S., Nam Y. J., Cho K.-J., Kim S. (2019) Research on technology status and development direction of wearable robot, *Fash.Text. Res. J.* **21**(5), 640–655.
- Knerr S., Personnaz L., Dreyfus G. (1990) Single-layer learning revisited: A stepwise procedure for building and training a neural network, *Neurocomputing*. Springer, pp. 41–50.

- Kramer O. (2013) K-nearest neighbors, *Dimensionality reduction with unsupervised nearest neighbors*. Springer, pp. 13–23.
- Krebel U.-G. (1999) Pairwise classification and support vector machines, *Advances in kernel methods: Support vector learning*, pp. 255–268.
- Li Z., Su C.-Y., Li G., Su H. (2014) Fuzzy approximation-based adaptive backstepping control of an exoskeleton for human upper limbs, *IEEE Trans. Fuzzy Syst.* **23**(3), 555–566.
- Liu X., Low K. (2004) *Development and preliminary study of the NTU lower extremity exoskeleton*. Paper presented at the IEEE Conference on Cybernetics and Intelligent Systems, 2004.
- Moreno J., Bueno L., Pons J. L., Baydal-Bertomeu J., Belda-Lois J., Prat J., Barberá R. (2008) Wearable robot technologies, *Wearable robots: Biomechatronic exoskeletons*. John Wiley & Sons, Hoboken, pp. 165–199.
- Murphy K. P. (2006) Naive bayes classifiers, *Univ. B. C.* **18**(60), 1–8.
- Murtagh F. (1991) Multilayer perceptrons for classification and regression, *Neurocomputing* **2**(5–6), 183–197.
- Pal S. K., Mitra S. (1992) Multilayer perceptron, fuzzy sets, classification.
- Pascanu R., Mikolov T., Bengio Y. (2013) *On the difficulty of training recurrent neural networks*. Paper presented at the International conference on machine learning.
- Peterson L. E. (2009) K-nearest neighbour, *Scholarpedia* **4**(2), 1883.
- Proietti T., Jarrassé N., Roby-Brami A., Morel G. (2015) *Adaptive control of a robotic exoskeleton for neurorerehabilitation*. Paper presented at the 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER).
- Qin X., Zhang W., Gao S., He X., Lu J. (2018) *Sensor fault diagnosis of autonomous underwater vehicle based on LSTM*. Paper presented at the 2018 37th Chinese Control Conference (CCC).
- Reaz M. B. I., Hussain M. S., Mohd-Yasin F. (2006) Techniques of EMG signal analysis: Detection, processing, classification and applications, *Biol. Proced. Online* **8**(1), 11–35.
- Reilly R. B., Lee T. C. (2010) Electrograms (ecg, eeg, emg, eog), *Technol. Health Care* **18**(6), 443–458.
- Rish I. (2001) *An empirical study of the naive Bayes classifier*. Paper presented at the IJCAI 2001 workshop on empirical methods in artificial intelligence.
- Safavian S. R., Landgrebe D. (1991) A survey of decision tree classifier methodology, *IEEE Trans. Syst. Man Cybern.* **21**(3), 660–674.
- Sakurai T., Sankai Y. (2009) *Development of motion instruction system with interactive robot suit HAL*. Paper presented at the 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO).
- Song Y.-Y., Ying L. (2015) Decision tree methods: Applications for classification and prediction, *Shanghai Arch. Psychiatry* **27**(2), 130.
- Subasi A., Ercelesi E. (2005) Classification of EEG signals using neural network and logistic regression, *Comput. Methods Programs Biomed.* **78**(2), 87–99.
- Vapnik V. N. (1999) An overview of statistical learning theory, *IEEE Trans. Neural Networks* **10**(5), 988–999.
- Vikas V., Crane III C. D. (2013) *Measurement of robot link joint parameters using multiple accelerometers and gyroscope*. Paper presented at the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.
- Wang Y.-L., Wang K.-Y., Wang W.-L., Yin P.-C., Han Z. (2019) Appraise and analysis of dynamical stability of cable-driven lower limb rehabilitation training robot, *J. Mech. Sci. Technol.* **33**(11), 5461–5472.
- Weston J., Watkins C. (1998) *Multi-class support vector machines*.
- Wu W., Rao S. (2007) Uncertainty analysis and allocation of joint tolerances in robot manipulators based on interval analysis, *Reliab. Eng. Syst. Saf.* **92**(1), 54–64.
- Yang G., Tan W., Jin H., Zhao T., Tu L. (2019) Review wearable sensing system for gait recognition, *Cluster Comput.* **22**(2), 3021–3029.
- Yeem S., Heo J., Kim H., Kwon Y. (2018) Technical analysis of exoskeleton robot, *World J. Eng. Technol.* **7**(1), 68–79.

- Zhang J., Li Q., Wang B., Cui M. (2020) *A novel sudden fault prediction method based on hierarchical structure with GRU neural network for X-ray high-voltage power supply*. Paper presented at the Journal of Physics: Conference Series.
- Zhao R., Wang D., Yan R., Mao K., Shen F., Wang J. (2017) Machine health monitoring using local feature-based gated recurrent unit networks, *IEEE Trans. Ind. Electron.* **65**(2), 1539–1548.
- Zhu C., Sun W., Sheng W. (2008) *Wearable sensors based human intention recognition in smart assisted living systems*. Paper presented at the 2008 International Conference on Information and Automation.

Chapter 7

Intelligent Manufacturing Performance Prediction and Application

7.1 Introduction

An intelligent manufacturing system consists of machines and buffers. Machines provide processing, transportation, or assembly services for parts. Typical machines include machine tools, machine centers, industrial robots, and mobile robots. Buffers supply temporary storage areas for parts waiting before machines, which makes parts flow smoothly. Because of the nonsynchronous processing time of parts in different machines, blocking and starving phenomena occur. Production disturbance may cause uncertain production halts. These cause a loss in system performance. One option to decrease this loss is to allocate additional buffers and utilize machines with high service rates. However, too many buffers increase system redundancy, leading to longer sojourn time and higher cost, whereas an extremely high service rate increases machine idle time and waste energy. Therefore, it is necessary to reasonably configure buffer capacities and machine service rates in intelligent manufacturing systems. This configuration problem is typically solved by iteratively generating candidate solutions and evaluating system performance corresponding to candidate solutions.

Previous studies have proposed many methods to evaluate system performance, including throughput rate, work-in-process, average waiting time, and system cost (Gao *et al.*, 2021; Gu *et al.*, 2020; Song and Moon, 2019). These economic indicators were important to evaluate an intelligent manufacturing system. Recently, with green manufacturing development, green indicators such as energy consumption and energy efficiency have increasingly become popular in demonstrating the advantage of an intelligent manufacturing system. Furthermore, traditional system performance evaluation methods consist of exact, approximation, and simulation methods (Weiss *et al.*, 2019; Demir *et al.*, 2014). Exact methods obtain results that are very close to actual values (Gordon and Newell, 1967). However, due to high computation complexity, they are typically suitable for an intelligent manufacturing system with small-scale. Simulation methods are based on discrete event simulation and can achieve high evaluation accuracy (Oljira *et al.*, 2020). However, long environment

settings and modeling time limit their application in system configuration. Approximation methods attempt to balance evaluation accuracy and efficiency (Yan *et al.*, 2021; Shin *et al.*, 2019). However, with the increasing efficiency requirement for intelligent manufacturing configuration and digital twin in the industry, a more efficient system performance evaluation needs to be studied. Currently, data fusion technology has been used in many research areas of robotics and manufacturing, such as bottleneck detection and life prediction (Wang *et al.*, 2022). The results have proven the effectiveness of data fusion technology and its application possibility in the process control of an intelligent manufacturing system. Therefore, system performance prediction, which is based on data fusion technology, has been proposed to evaluate system performance efficiently. Figures 7.1 and 7.2 show the method and indicator classification of performance evaluation, respectively.

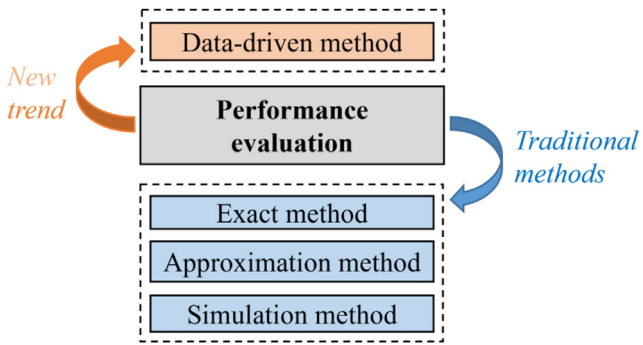


FIG. 7.1 – Classification of performance evaluation methods.

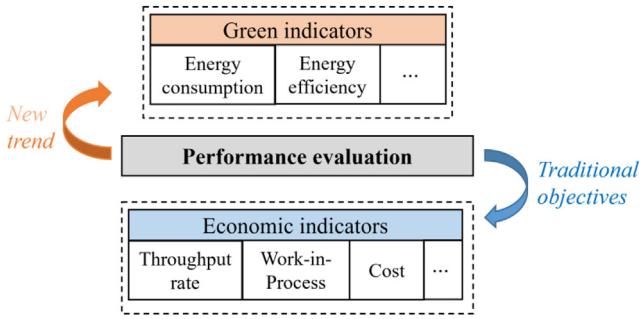


FIG. 7.2 – Classification of performance evaluation indicators.

In this chapter, performance prediction for an intelligent manufacturing system is introduced. In addition, the application of performance prediction in intelligent manufacturing systems has been presented. The research flowchart of system performance prediction is divided into four parts: data acquisition, prediction modeling, application, and conclusion. The research framework is shown in figure 7.3.

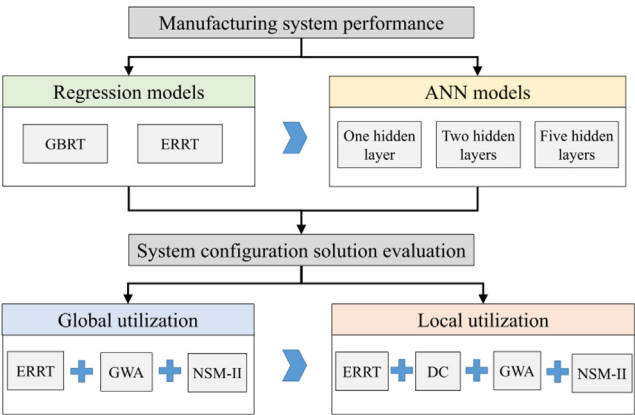


FIG. 7.3 – Research framework.

7.2 Data Acquisition

Manufacturing system performance prediction aims at analyzing the mapping relationships between decision variable data and objective performance data, as is shown in figure 7.4. Common decision variable data include buffer allocation vector which denotes buffer capacities, service rate allocation vector which denotes machine processing time, failure rate vector which denotes machine failure frequency, and repair rate vector which denotes machine maintenance frequency. Common objective performance data include throughput rate, work-in-process, average waiting time, cost, energy consumption, and energy efficiency. According to different intelligent manufacturing systems’ characteristics, working environments, and design objectives, decision variables and objective performance change.

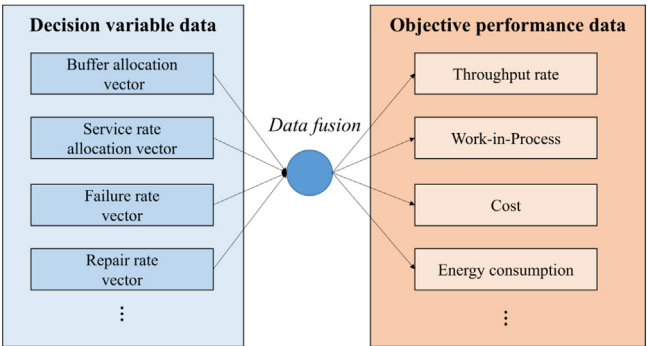


FIG. 7.4 – Decision variable data and objective performance data.

7.2.1 Data-Driven Method

There are two methods to collect decision variable data and objective performance data. One option is to conduct experiments for an intelligent manufacturing system. In a sufficient experiment time, industrial data are collected using Product-Embedded Information Devices (PIED) according to different manufacturing system states. Figure 7.5 shows the architecture of the data-driven method. After raw data pre-processing, the collected data can be used to build prediction models. Previous studies have utilized the data-driven method to collect manufacturing system data and achieve the throughput rate prediction with acceptable accuracy (Tsadiras *et al.*, 2013). The data-driven method can collect industrial data that are close to actual system performance, contributing to high predictive accuracy. However, experiment time and cost are expensive. Furthermore, with the working environment changing significantly, the operation process of an intelligent manufacturing system also changes, leading to previous data becoming invalid. These limit the application of the data-driven method.

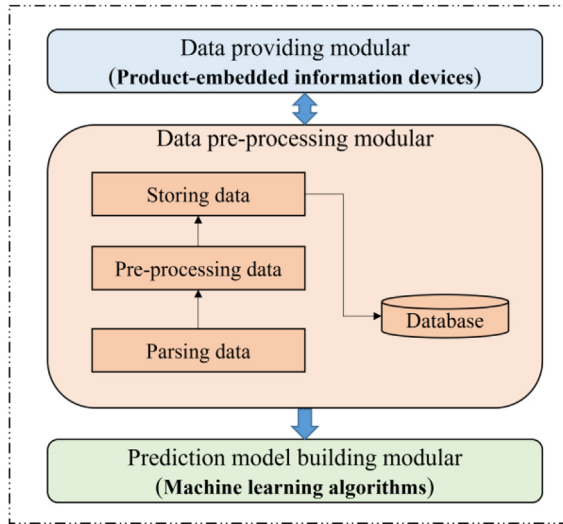


FIG. 7.5 – Data-driven method.

7.2.2 Model-Driven Method¹

The model-driven method builds the simulation or analytical model of an intelligent manufacturing system and generates data using the model. Simulation models can

¹Part of text in sections 7.2.2–7.4.1 is re-used from Sage Publication article “A data-driven ensemble algorithm of black widow optimizer and simulated annealing algorithms for multi-objective buffer allocation in production lines” (Gao and Liu, 2023) under permission.

be built using discrete event simulation software such as Quest, Flexsim, Anylogic, and Arena (Mohammad and Ismail, 2021; Oljira *et al.*, 2020). Analytical models are typically built based on queueing and probability theory. Because decision variable and objective performance data are generated by a simulation or an analytical model, no pre-processing is needed. Unlike the data-driven method, the model-driven method can collect data with less time and cost. However, simulation and analytical model accuracy are challenging, affecting generated data validity significantly. Figure 7.6 shows the architecture of the model-driven method.

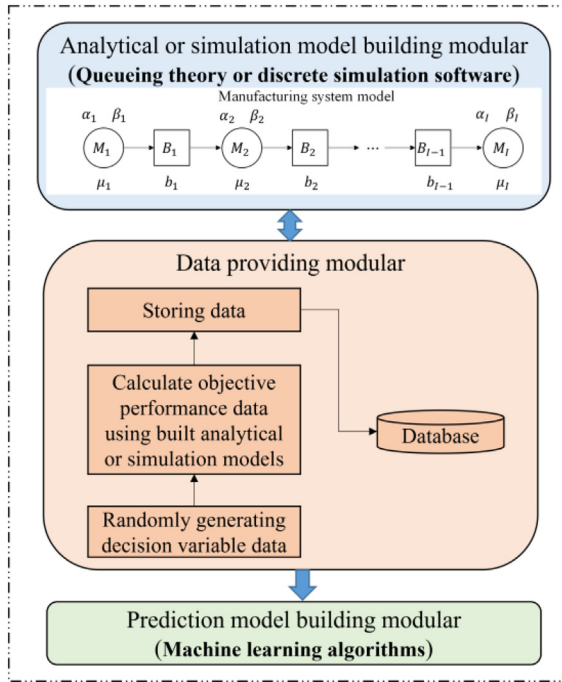


FIG. 7.6 – Model-driven method.

In this section, an analytical model is built. The model generates data for an intelligent manufacturing system to show the model-driven mechanism. In figure 7.6, a manufacturing system model is presented. Squares represent buffers, circles represent machines and arrows represent part flow direction. There are I machines and $I - 1$ buffers in the manufacturing system. b_i denotes the capacity of buffer i . μ_i denotes the service rate (reciprocal of processing time) of machine i . α_i and β_i denote the failure rate and repair rate of machine i , respectively. According to the approximation method in the literature (Alaouchiche *et al.*, 2020), the analytical model of a manufacturing system can be built. Alaouchiche's method calculates effective service rates of machines based on system steady-state probabilities. This process continues until all effective service rates converge. The system throughput

rate is the lowest effective service rate among machines. For each buffer, its steady-state probabilities can be expressed as follows:

$$p^i = \begin{cases} \frac{\gamma_i \times (1 - \gamma_i)}{1 - \gamma_i^{K_i+1}}, & \gamma_i \neq 1 \\ \frac{1}{K_i+1}, & \gamma_i = 1 \end{cases} \quad (7.1)$$

where K_i denotes the buffer capacity of buffer i . γ_i denotes the service rate ratio of buffer i . γ_i can be calculated by

$$\gamma_i = \frac{\min_{k=1 \dots i} \rho_k}{\min_{k=i+1 \dots I} \rho_k} \quad (7.2)$$

The steady-state probabilities of empty and full buffer capacity can be expressed as follows:

$$p_i^0 = \begin{cases} \frac{1 - \gamma_i}{1 - \gamma_i^{K_i+1}}, & \gamma_i \neq 1 \\ \frac{1}{K_i+1}, & \gamma_i = 1 \end{cases} \quad (7.3)$$

$$p_i^{K_i} = \begin{cases} \frac{\gamma_i^{K_i} \times (1 - \gamma_i)}{1 - \gamma_i^{K_i+1}}, & \gamma_i \neq 1 \\ \frac{1}{K_i+1}, & \gamma_i = 1 \end{cases} \quad (7.4)$$

The effective service rate of the machine i can be expressed as follows:

$$\rho_i = \mu_i \times \frac{\beta_i \times \xi_i}{\beta_i \times \xi_i + \alpha \times \xi_i}, \quad i = 1, \dots, I \quad (7.5)$$

$$\begin{cases} \xi_1 = 1 - p_1^{K_1} \\ \xi_I = 1 - p_{I-1}^0 \\ \xi_i = (1 - p_{i-1}^0) \times (1 - p_i^{K_i}), \quad i = 2, \dots, I-2 \end{cases} \quad (7.6)$$

By solving equations (7.1)–(7.6), the throughput rate can be expressed as follows:

$$\theta = \min_{i=1, \dots, I} \{\rho_i\} \quad (7.7)$$

Alaouchiche's method calculates the energy consumption of each machine (E_i) based on its failure rates (α_i), repair rates (β_i), steady-state probabilities of the buffer in front of it ($p_i^{K_i}$), and energy consumption of different operation states. The system energy consumption is the sum of the energy consumption of I machines as equation (7.8). Readers can refer to the literature (Alaouchiche *et al.*, 2020) for the details of the throughput and energy consumption calculation.

$$EC = \sum_{i=1}^I E_i \quad (7.8)$$

After the analytical model is built based on Alaouchiche's method, decision variable data are generated randomly. Then, the analytical model can calculate their objective performance data.

7.3 Prediction Modeling

Performance prediction is related to the regression problem. Two types of algorithms are used to solve the problem: regression algorithms and Artificial Neural Networks (ANN). Classical regression algorithms consist of linear regression, polynomial regression, support vector machine, decision tree, and random forest. Recently, more robust methods such as the Gradient Boosting Regression Tree (GBRT) and the Extremely Randomized Regression Tree (ERRT) have become popular. This section introduces two different methods for building prediction models of an intelligent manufacturing system, including the GBRT, ERRT, and ANN.

7.3.1 Regression Algorithms

The GBRT and ERRT are popular machine-learning algorithms that effectively solve classification and regression problems (Ke *et al.*, 2017; Geurts *et al.*, 2006). The GBRT ensembles regression trees as weak learners to generate a power learner. Weights are iteratively added to the weak learners based on the prediction results in the power learner to improve the final results. This process continues until a convergence criterion is met. In addition, the ERRT builds an ensemble of the unpruned decision or regression trees according to the classical top-down procedure. Unlike the random forest, the ERRT splits nodes by choosing cut-points fully at random and uses the whole learning sample to grow the trees. Compared to single algorithms—such as the PR and DT—it is easier for the GBRT and ERRT to achieve high prediction accuracy and robustness for complex regression problems (Si and Du, 2020). Consequently, this section introduces these two ensemble algorithms.

Furthermore, previous scholars proposed the Grey Wolf Algorithm (GWA) to efficiently search for nearby optimal solutions to optimization problems (Nadimi-Shahraki *et al.*, 2021; Mirjalili *et al.*, 2014, 2016). Therefore, the GWA was used to optimize the hyperparameters in the GBRT and ERRT to improve its predictive accuracy.

(1) Gradient Boosting Regression Tree (GBRT)

A. Modeling

The prediction model based on the GBRT can be expressed as follows:

$$F^j(\bar{x}) = \hat{y}_q^j = \hat{y}_q^{j-1} + f_j(\bar{x}_q) \quad (7.9)$$

where $F^j(\bar{x})$ is the function to obtain $\hat{y}_q^j$ that is the vector of predictive values, $\bar{x}_q$ is the input sample, q is the number of data samples, and j is the number of trees or

iterations. At iteration j , the current prediction $\hat{y}_q^j$ can be improved using the previous prediction $\hat{y}_q^{j-1}$ and the current estimator $f_j(\bar{x}_q)$.

B. Performance metrics

For a prediction model, it is necessary to evaluate the predictive accuracy. For each predictive objective, the Mean Square Error (MSE), Mean Absolute Error (MAE), and decision factor (R^2) are evaluated. Their average values are the final evaluation measures. The MSE can be expressed as follows:

$$\sigma_\theta = \frac{1}{Q_{\text{tes}}} \sum_{q=1}^{Q_{\text{tes}}} (\theta_q - \hat{\theta}_q)^2 \quad (7.10)$$

$$\sigma_{\text{EC}} = \frac{1}{Q_{\text{tes}}} \sum_{q=1}^{Q_{\text{tes}}} (\text{EC}_q - \overline{\text{EC}}_q)^2 \quad (7.11)$$

$$\sigma = \frac{\sigma_\theta + \sigma_{\text{EC}}}{2} \quad (7.12)$$

where q is a testing sample and Q_{tes} is the number of testing samples; $\hat{\theta}_q$ and $\overline{\text{EC}}_q$ are the throughput rate and energy consumption predicted by the prediction models, respectively; θ_q and EC_q are the actual throughput rate and energy consumption determined from the testing samples. The MAE can be expressed as follows:

$$\varepsilon_\theta = \frac{1}{Q_{\text{tes}}} \sum_{q=1}^{Q_{\text{tes}}} |\theta_q - \hat{\theta}_q| \quad (7.13)$$

$$\varepsilon_{\text{EC}} = \frac{1}{Q_{\text{tes}}} \sum_{q=1}^{Q_{\text{tes}}} |\text{EC}_q - \overline{\text{EC}}_q| \quad (7.14)$$

$$\varepsilon = \frac{\varepsilon_\theta + \varepsilon_{\text{EC}}}{2} \quad (7.15)$$

and R^2 can be expressed as follows:

$$R_\theta^2 = 1 - \frac{\sum_{q=1}^{Q_{\text{tes}}} (\theta_q - \hat{\theta}_q)^2}{\sum_{q=1}^{Q_{\text{tes}}} (\theta_q - \theta_{\text{mea}})^2} \quad (7.16)$$

$$R_{\text{EC}}^2 = 1 - \frac{\sum_{q=1}^{Q_{\text{tes}}} (\text{EC}_q - \widehat{\text{EC}}_q)^2}{\sum_{q=1}^{Q_{\text{tes}}} (\text{EC}_q - \text{EC}_{\text{mea}})^2} \quad (7.17)$$

$$R^2 = \frac{R_\theta^2 + R_{EC}^2}{2} \quad (7.18)$$

where θ_{mea} and EC_{mea} are the mean values in the testing samples. The MSE, MAE, and R^2 values should all be in the 0–1 range. Furthermore, low MSE and MAE values and high R^2 values indicate better predictive performance.

C. Numerical example

Setting

It is assumed that a manufacturing system involves $I - 1$ buffers and I machines. Its total buffer capacity is set to K which is the sum of each buffer capacity. Its total service rate is set to U which is the sum of each machine's service rate. Here, decision variable data consists of a buffer allocation vector ($\bar{b}$) and service rate allocation vector ($\bar{\mu}$). Objective performance data consist of throughput rate (θ) and Energy Consumption (EC). Decision variable data are generated randomly, and objective performance data are calculated based on the analytical model introduced in section 7.2.2. Ten thousand samples were selected of which 8000 samples were used to train prediction models, and 2000 samples were used to verify them. The numerical example setting is listed in table 7.1.

In addition, to compare the efficiency between prediction models and the analytical model. From Patterns 1 to 3, the prediction models based on the GBRT and the analytical model built in section 7.2.2 were used to calculate 10 000 samples to show the computation time gap between the two kinds of models.

TAB. 7.1 – Numerical example setting.

Pattern number (I, K, U)	Decision variables	Objective performance	Number of samples
1 (5, 16, 5)	$\bar{b}, \bar{\mu}$	θ, EC	10 000
2 (5, 20, 5)			
3 (5, 16, 7.5)			

Results

Table 7.2 summarizes the MAE, MSE, and R^2 values of the prediction models based on the GBRT. Figure 7.7 compares the predictive and actual values for Patterns 1, 2, and 3, respectively. Table 7.3 compares the computation time of the prediction models based on the GBRT and the analytical model built in section 7.2.2. The following conclusions can be drawn:

The MAE and MSE are close to 0, whereas the R^2 is close to 1, indicating the high predictive accuracy of the GBRT. Consequently, most points are on the line through the origin in figure 7.7.

- (a) From Patterns 1 to 3, total buffer capacity and total service rate increase, enlarging the range of decision variable data and increasing the regression-fitting

- complexity. Consequently, the MAE and MSE increase and the R^2 decreases, indicating the predictive accuracy decreasing.
- (b) Compared to the total service rate, the increase in total buffer capacity generates a wider range of decision variable data. Consequently, the predictive accuracy decreases more apparently in Patten 2 than in Patten 3.
 - (c) Due to discrete buffer capacities and service rates, the system performance values are distributed dispersedly. Therefore, some points are not continuous in figure 7.7a.
 - (d) The prediction models can decrease computation time significantly compared to the analytical model. Furthermore, if a performance prediction model is used in the system configuration of a manufacturing system, which requires a large number of evaluations, the efficiency advantage of the prediction model will be more apparent.

TAB. 7.2 – MAE, MSE, and R^2 values of the prediction models based on the GBRT.

Pattern number (I, K, U)	MAE (%)	MSE (%)	R^2
1 (5, 16, 5)	0.0254	0.0044	1
2 (5, 20, 5)	0.0308	0.0060	1
3 (5, 16, 7.5)	0.0263	0.0043	1

(2) *Extremely Randomized Regression Tree (ERRT)*

A. *Modeling*

The ERRT fitting procedure can be expressed as follows: assuming several decision trees are used as weaker learners, the ERRT iteratively ensembles the predictive values of the tree and aggregates the final predictive value by arithmetic average. In every iteration, the ERRT generates a number of attributes randomly selected at each node and decides the minimum sample size for splitting a node. By this method, the weights of trees in the ERRT can be expressed.

B. *Performance metrics*

The MAE, MSE, and R^2 are used to evaluate predictive accuracy.

C. *Numerical example*

Setting

The settings are the same as in table 7.1.

Results

Table 7.4 summarizes the MAE, MSE, and R^2 values of the prediction models based on the ERRT. Figure 7.8 compares the predictive and actual values for Patterns 1, 2, and 3, respectively. The following conclusions can be drawn:

- (a) The MAE and MSE are close to 0, whereas the R^2 is close to 1, indicating the high predictive accuracy of the ERRT. Consequently, most points are on the line through the origin in figure 7.8.

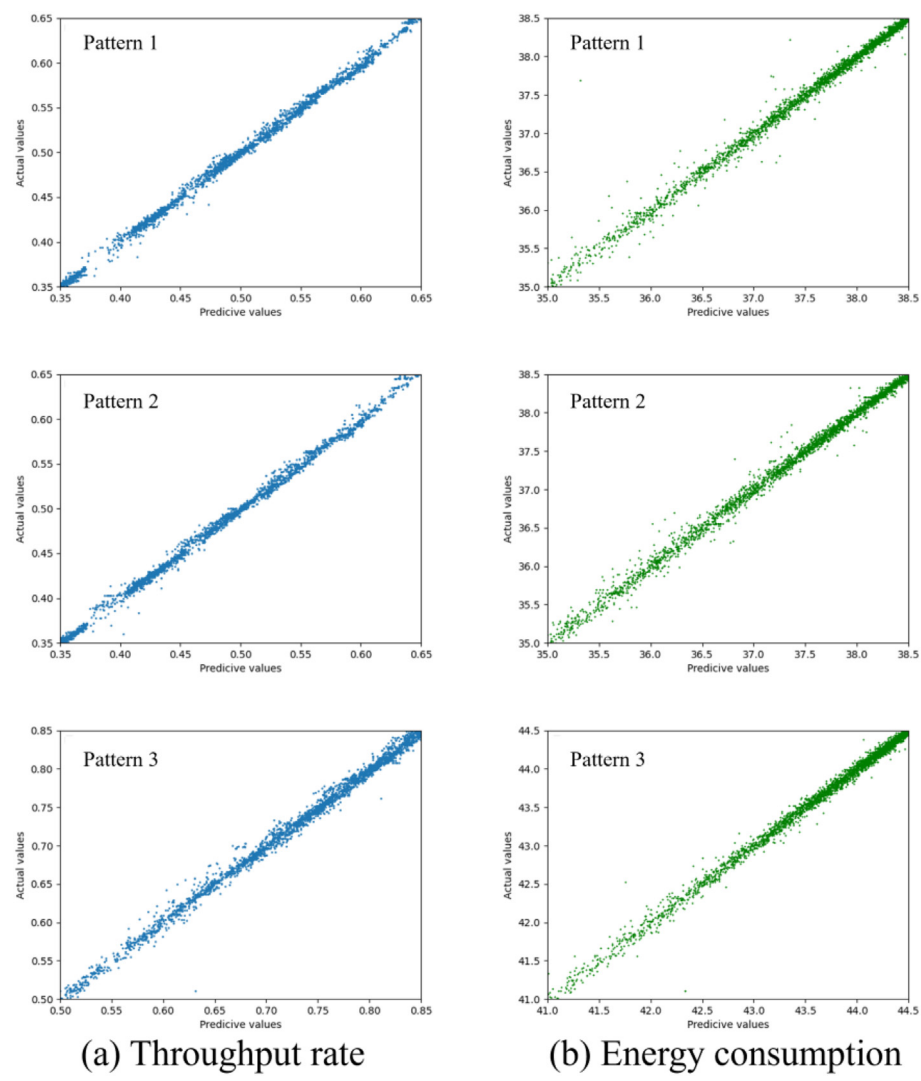


FIG. 7.7 – Comparison of predictive values and actual values of the prediction models based on the GBRT.

TAB. 7.3 – Computation time of the prediction models and the analytical model.

Pattern number (I, K, U)	Prediction models	The analytical model
1 (5, 16, 5)	2.97 s	45.99 s
2 (5, 20, 5)	4.10 s	56.45 s
3 (5, 16, 7.5)	3.66 s	50.31 s

TAB. 7.4 – MAE, MSE, and R^2 values of the prediction models based on the ERRT.

Pattern number (I, K, U)	MAE (%)	MSE (%)	R^2
1 (5, 16, 5)	0.0122	0.0036	1
2 (5, 20, 5)	0.0177	0.0047	1
3 (5, 16, 7.5)	0.0129	0.0028	1

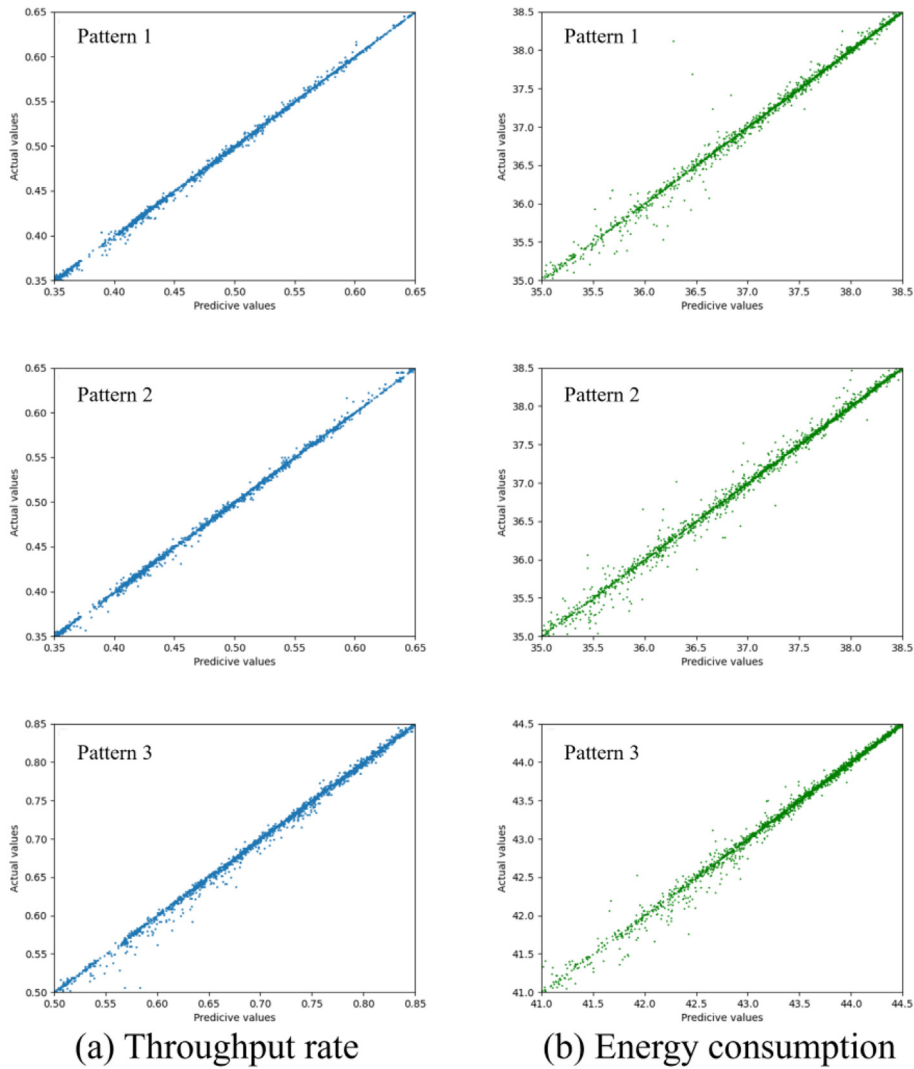


FIG. 7.8 – Comparison of predictive values and actual values of the prediction models based on the ERRT.

- (b) From Patterns 1 to 3, total buffer capacity and total service rate increase, enlarging the range of decision variable data and increasing the regression-fitting complexity. Consequently, the MAE and MSE increase and the R^2 decreases, indicating the predictive accuracy decreasing.
- (c) Compared to the total service rate, the increase in total buffer capacity generates a wider range of decision variable data. Consequently, the predictive accuracy decreases more apparently in Patten 2 than in Patten 3.
- (d) Compared to the GBRT, the ERRT achieves better predictive accuracy, indicating the advantage of the ensemble method in the ERRT for the performance prediction of an intelligent manufacturing system.

7.3.2 Artificial Neural Network (ANN)

A. Modeling

The ANN is a classical machine learning algorithm to solve regression problems. Tsadiras *et al.* (2013) proposed an ANN model with a hidden layer to predict the performance of manufacturing systems. Huang *et al.* (1999) believed that one or two hidden layers generate low predictive errors in performance prediction. Consequently, ANN models with one and two hidden layers are used to build prediction models. An ANN with five hidden layers, which represents the ANN with multiple hidden layers, is also compared to verify the sensitivity of the hidden layer size. To improve the predictive performance, the GWA was also used to optimize the neuron size of each hidden layer. Figure 7.9 shows the ANN structure with two hidden layers in performance prediction.

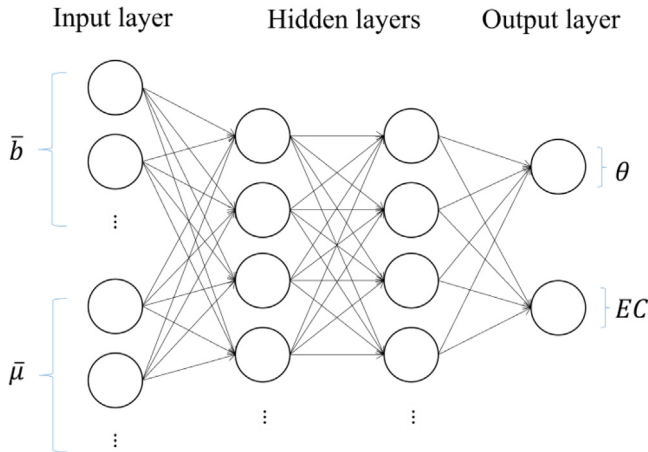


FIG. 7.9 – ANN structure with two hidden layers.

B. Performance metrics

The MAE, MSE, and R^2 are used to evaluate predictive accuracy.

C. Numerical example

Setting

The settings are the same as in table 7.1. In addition, to test the sensitivity of the hidden layer size in the ANN for its predictive accuracy. The ANNs with one, two, and five hidden layers have been tested in this section.

Results

Table 7.5 summarizes the MAE, MSE, and R^2 values of the prediction models based on the ANN with one, two, and five hidden layers. Figures 7.10–7.12 compare the predictive and actual values for the ANN with one, two, and five hidden layers, respectively. The following conclusions can be drawn:

- (a) The MAE and MSE are close to 0. However, the R^2 is unsatisfactory. Therefore, some points are away from the line through the origin in figures 7.10–7.12, especially for the throughput rate.
- (b) From Patterns 1 to 3, with the increase of the total buffer capacity and total service rate, the MAE and MSE increase and the R^2 decreases, indicating the predictive accuracy decreasing.
- (c) Compared to the total buffer capacity, the increase in the total service rate affects the predictive accuracy of the ANN more significantly, leading to accuracy decreasing more apparently in Pattern 3 than in Pattern 2.
- (d) The predictive accuracy of the prediction models based on the ANNs with one and two hidden layers is close.
- (e) The predictive accuracy of the throughput rate for the ANN with two hidden layers is better than the ANN with one hidden layer. However, the ANN with one hidden layer achieves better accuracy for the energy consumption prediction.
- (f) The predictive accuracy for the throughput rate is lower than that for energy consumption in figures 7.10 and 7.11. However, the unbalance between the predictive accuracy of the two outputs becomes small. Therefore, figures 7.12a and 7.12b become similar.

TAB. 7.5 – MAE, MSE, and R^2 values of the prediction models based on the ANN.

Pattern number (I, K, U)	Hidden layer size	MAE (%)	MSE (%)	R^2
1 (5, 16, 5)	1	0.0115	0.0051	0.99
	2	0.0125	0.0055	0.99
	5	0.0188	0.0070	0.99
2 (5, 20, 5)	1	0.0121	0.0053	0.99
	2	0.0222	0.0080	0.99
	5	0.0219	0.0073	0.99
3 (5, 16, 7.5)	1	0.0385	0.0060	0.99
	2	0.0458	0.0085	0.99
	5	0.0845	0.0208	0.98

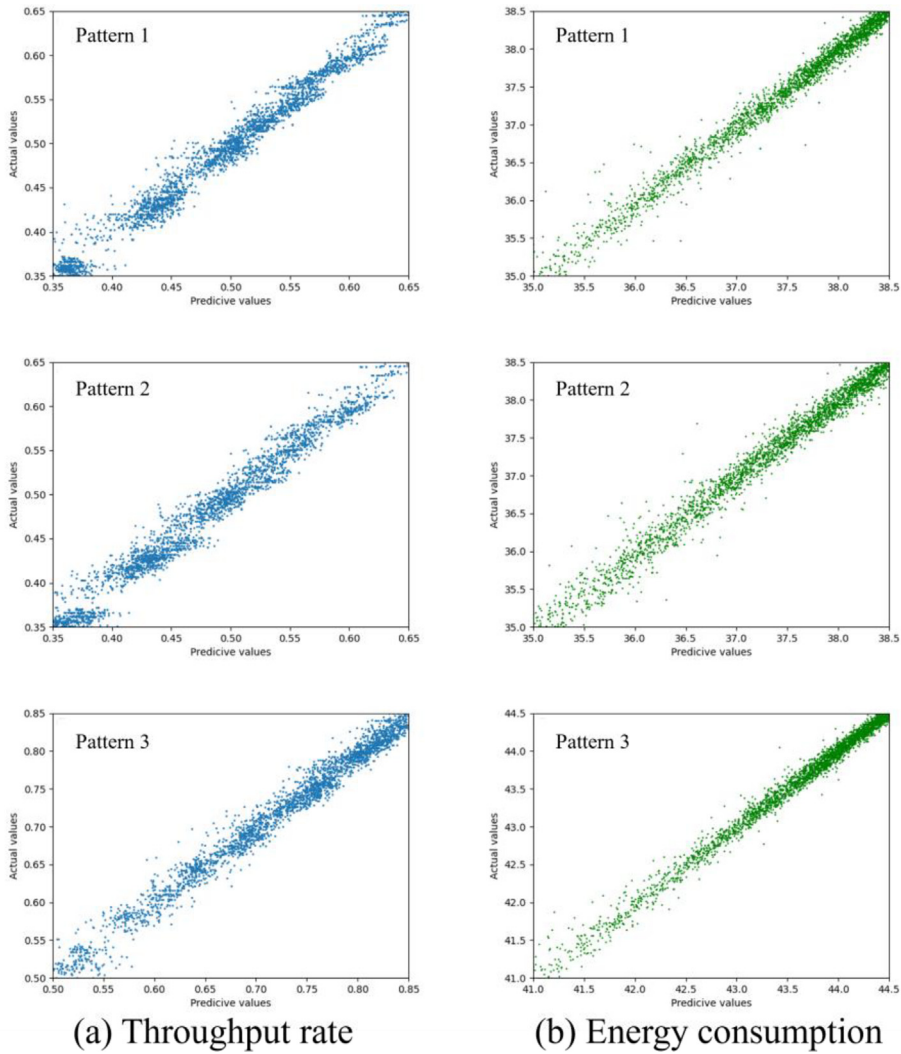


FIG. 7.10 – Comparison of predictive values and actual values of the prediction models based on the ANN with one hidden layer.

- (g) With the increase of the hidden layers, the MAE and MSE increase and R^2 decreases, indicating that the predictive accuracy of the prediction models decreases.
- (h) The hidden layer size hardly improves predictive accuracy; however, it can balance predictive accuracy among multiple outputs.
- (i) The ANNs with one or two hidden layers are more suitable for building prediction models of manufacturing system performance.

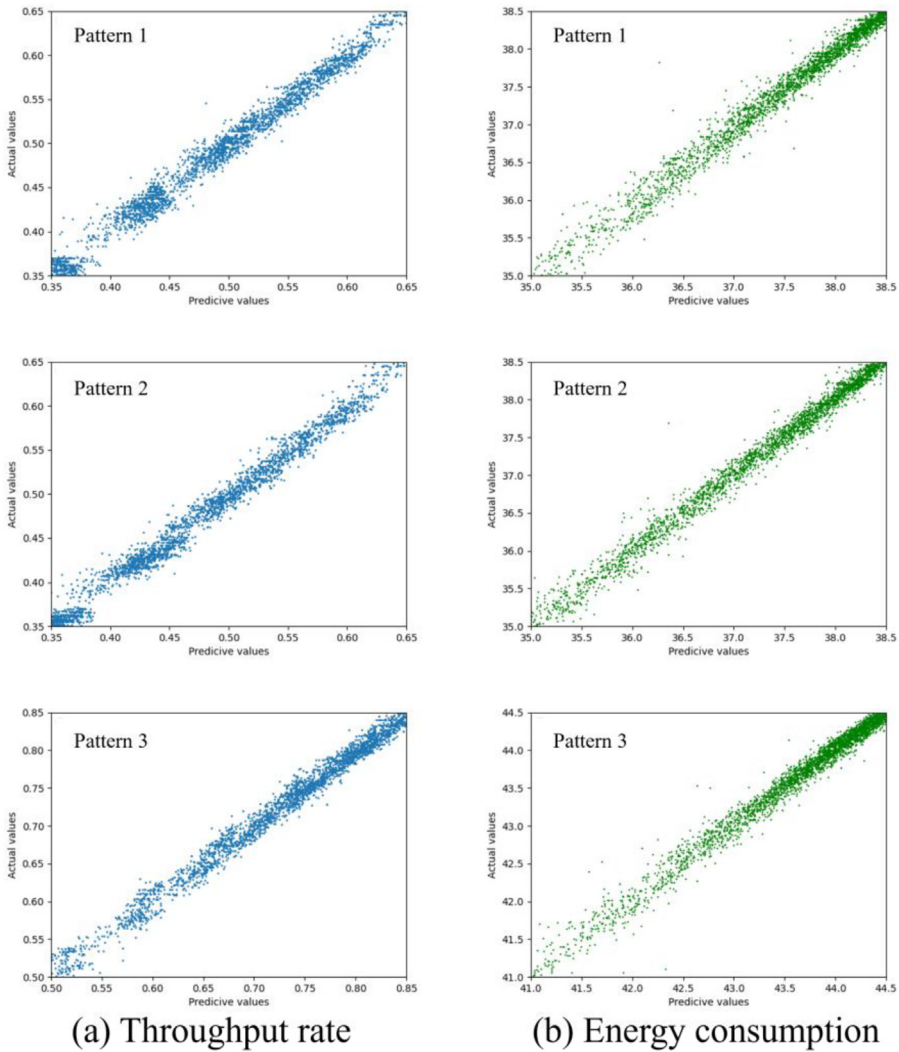


FIG. 7.11 – Comparison of predictive values and actual values of the prediction models based on the ANN with two hidden layers.

7.3.3 Comparison Analysis

Sections 7.3.1 and 7.3.2 introduce prediction models based on regression algorithms and ANN. By comparing their fitting performance for Patterns 1, 2, and 3, the following conclusion can be drawn:

- (a) Although the ANN has been widely used in solving prediction problems, the ensemble algorithms can achieve better predictive accuracy in the performance prediction of intelligent manufacturing systems at this stage. In addition, the fitting efficiency of the ensemble algorithms is also satisfactory.

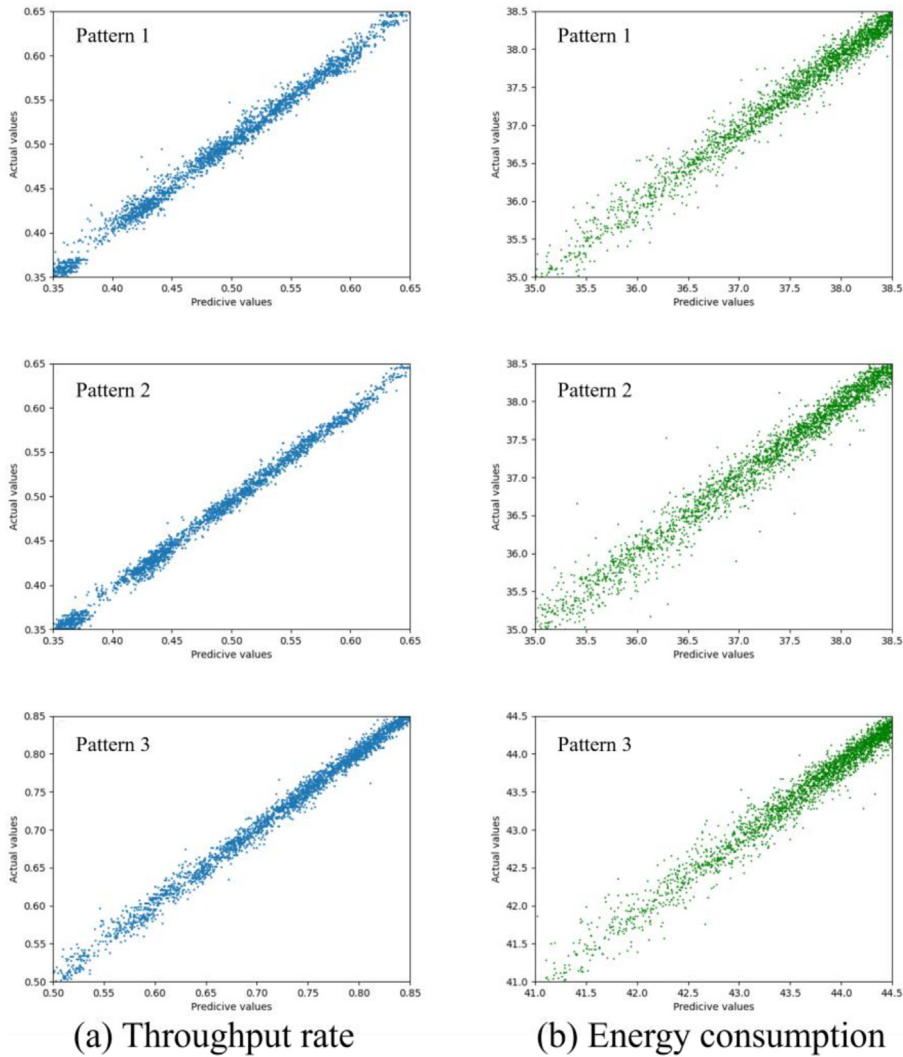


FIG. 7.12 – Comparison of predictive values and actual values of the prediction models based on the ANN with five hidden layers.

- (b) The ensemble algorithms utilize a weak learner and ensemble many weak learners to become as strong a learner. In the fitting process, the ensemble algorithms iteratively analyze the results of weaker learners, gradually decreasing predictive errors. The results demonstrate the advantage of ensemble learning in the performance prediction of a manufacturing system.
- (c) Compared to the ensemble algorithms, the predictive accuracy of the ANN is more sensitive to the increase in total buffer capacity and total service rate.
- (d) The ANNs with one or two hidden layers are more suitable for building a performance prediction model of intelligent manufacturing systems.

7.4 Application

7.4.1 System Configuration

One important application of performance prediction is to solve the System Configuration Problem (SCP). The SCP aims at configuring system resources to satisfy the predefined objectives subject to design constraints, including processing, storage, and transportation resources. For instance, the Buffer Allocation Problem (BAP) is an SCP, which allocates proper capacities to different buffers to improve system performance. The total buffer capacity is a constraint for the BAP. The BAP in this section is defined as finding $\bar{b} = (b_1, b_2, \dots, b_{I-1})$ to maximize θ and minimize EC, subject to $\sum_{i=1}^{I-1} b_i = K$ and $b_i \geq 1$, where $\bar{b}$ denotes the buffer allocation vector; b_i denotes the capacity in buffer i ; θ denotes throughput rate; EC denotes energy consumption; K denotes total buffer capacity; I denotes the number of machines in a manufacturing system.

In the BAP, performance prediction is used to evaluate candidate buffer allocation solutions, whereas generative methods are used to generate candidate buffer allocation solutions. Generative methods consist of enumeration methods, search methods, metaheuristics, and dynamic programming. Enumeration methods can obtain a global optimal solution in candidate buffer allocation solutions (Weiss *et al.*, 2019), while dynamic programming typically generates solutions that are very close to the global optimal solution (Sadr and Malhame, 2004). However, long computation time limits their application in the BAP of large-scale manufacturing systems. Search methods and metaheuristics attempt to balance solution quality and computation time (Xi *et al.*, 2022; Demir and Koyuncuoğlu, 2021). They obtain near optimal solutions for the BAP in acceptable computation time, even for a large-scale manufacturing system. Search methods update candidate solutions based on predefined guidance information, leading to local optimal solutions in some cases. Consequently, current studies typically integrated a performance evaluation method with metaheuristics to solve the BAP. This section introduces the performance prediction application by integrating it with metaheuristics to solve the BAP.

Furthermore, there are two kinds of applications for performance prediction in the BAP, including global utilization and local utilization.

(1) Global utilization

For global utilization, performance prediction is integrated with a generative method, such as a metaheuristic to solve the BAP. Figure 7.13 shows the basic flow of global utilization. In the solution flow, generative methods iteratively provide candidate solutions; meanwhile, the prediction models evaluate the candidate solutions to search for better ones. This process continues until stopping criteria are met. In this section, the prediction models are integrated with the GWA to solve the BAP, presenting the global utilization of performance prediction.

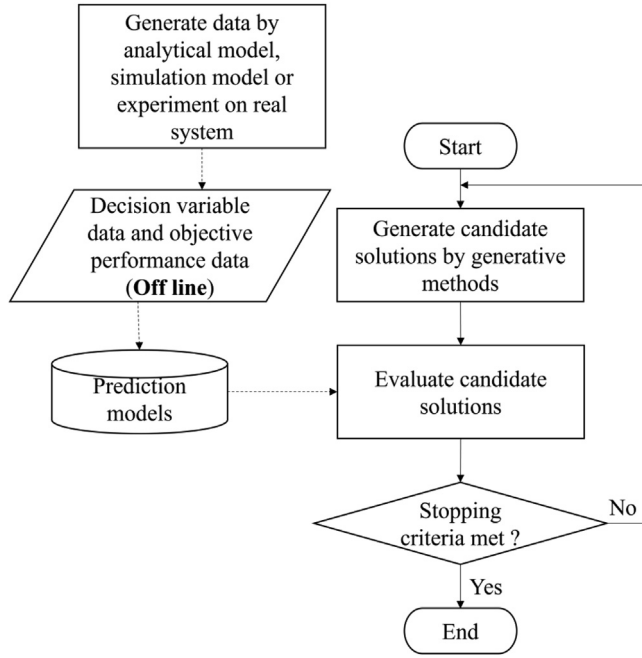


FIG. 7.13 – Basic flow of global utilization for performance prediction.

A. Gray Wolf Algorithm (GWA) in the BAP

The GWA was widely used in single-objective and multi-objective optimization problems. This section improved the GWA and integrated it with nondominated sorting method II (NSM-II) (K. Deb *et al.*, 2002) to solve the multi-objective BAP. The pseudocode of the Nondominant-Sorting-Gray-Wolf-Algorithm (NSGWA) is provided. The prediction models are utilized to predict system performance and evaluate candidate buffer allocation solutions. Table 7.6 lists the notion for the NSGWA.

Algorithm: the NSGWA to solve the multi-objective BAP

Initialize:

Generate N_{DB} initial solutions and their corresponding throughput and energy consumption in the database DB;

Build prediction models based on database DB;

Randomly generate N_{ini} initial solutions and append them to the archive Π_{arc} .

1	Set $j_{GWA} = 0$.
2	while $j_{GWA} < J_{GWA}$ do
3	$j_{GWA} = j_{GWA} + 1$
4	Generate neighborhood solutions of solutions in Π_{arc} using equations (7.19)–(7.23).
5	Revise buffer capacities in the neighborhood solutions by equation (7.24).
6	Normalize the neighborhood solutions by equation (7.25).

(continued).

Algorithm: the NSGWA to solve the multi-objective BAP	
7	Append the neighborhood solutions to Π_{nei} .
8	Give floor rank and calculate the crowding distance of the solutions in Π_{arc} and Π_{nei} .
9	Select N_{ini} solutions and update Π_{arc} based on the NSM-II.
10	Update Π_{bes} by the solutions in FL_1 .
11	end while
12	Output Π_{bes} .

TAB. 7.6 – Notion in the GWA for the multi-objective BAP.

Items	Description
Π_{arc}	Achive to store current buffer allocation solutions in each iteration
Π_{nei}	Achive to store neighborhood buffer allocation solutions
Π_{bes}	Achive to store buffer allocation solutions in the current Pareto optimal front
j_{GWA}	An iteration. j_{GWA} is the maximum number of iterations in the GWA
FL_1	Current Pareto optimal front
N_{ini}	The number of buffer allocation solutions in Π_{arc}
N_{DB}	The number of buffer allocation solutions and their corresponding system performance in the database which is used to build prediction models
i	A machine. I is the number of machines in a manufacturing system
b_i	A buffer capacity. $\bar{b} = (b_1, b_2, \dots, b_{I-1})$
μ_i	A service rate. $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_{I-1})$
K	Total buffer capacity
θ	Throughput rate
EC	Energy consumption

Genetic operator

In the NSGWA, two strategies of solution update are used in the genetic operator, imitating the prey behaviors of grey wolves (Mirjalili *et al.*, 2014). The first strategy mimics the behavior of encircling prey and can be expressed as follows:

$$b_i(j_{GWA} + 1) = b_i^1(j_{GWA}) - A \times D \quad (7.19)$$

$$D = |G \times b_i^1(j_{GWA}) - b_i(j_{GWA})| \quad (7.20)$$

$$A = 2 \times a \times \text{rand}(0, 1) - a \quad (7.21)$$

$$G = 2 \times \text{rand}(0, 1) \quad (7.22)$$

where $b_i^1(j_{\text{GWA}})$ denotes the buffer capacity in the first selected solution in the current iteration; a is linearly decreased from 2 to 0 over iterations; A , C , and D are process parameters; $\text{rand}(0, 1)$ denotes a random number between 0 and 1. The second strategy is the behavior of hunting and can be expressed as follows:

$$b_i(j_{\text{GWA}} + 1) = \frac{b_i^1(j_{\text{GWA}}) + b_i^2(j_{\text{GWA}}) + b_i^3(j_{\text{GWA}})}{3} \quad (7.23)$$

where $b_i^1(j_{\text{GWA}})$, $b_i^2(j_{\text{GWA}})$ and $b_i^3(j_{\text{GWA}})$ are buffer capacities in the first, second, and third selected solutions in the current iteration, respectively. The chosen solutions are obtained by randomly selecting a solution from the higher part of the sorted solutions (the highest 10%, 20%, and 30% of the sorted solutions, respectively), consisting of the less crowded solutions. This selection strategy has been proven to be effective in the literature (Got *et al.*, 2020).

To avoid the buffer capacity being lower than the lower limit, the reflection operator is expressed as

$$b_i(j_{\text{GWA}} + 1) = b_{\min} \quad (7.24)$$

where $b_{\min}$ is the lower limit of the buffer capacity. Furthermore, to maintain the total buffer capacity of new solutions, the normalization operator is expressed as

$$b_i(j_{\text{GWA}} + 1) = \text{round}(K \times b_i(j_{\text{GWA}} + 1)) / \sum_{i=1}^{I-1} b_i(j_{\text{GWA}} + 1) \quad (7.25)$$

Floor rank and crowding distance calculation

Then Pareto dominance relationships among the solutions are detected, and their floors are ranked. The Pareto optimal front belongs to the first floor. Excluding all Pareto optimal solutions, the remaining solutions are ranked again based on the Pareto dominance relationship to obtain the following floors until all the solutions are ranked. For the solutions on the same floor, their crowding distances are used to classify their priorities. The solution with a high crowding distance has a higher priority, indicating a better solution. Readers can refer to the literature (K. Deb *et al.*, 2002) for more details about the NSM-II.

Stopping criterion

Commonly used stopping criterion in metaheuristics include arriving at the allowed maximum iterations and satisfying predefined objectives. In this study, the stopping criterion for the NSGWA is defined as the maximum number of iterations (J_{GWA}).

B. Performance metrics

The number of solutions in the Pareto optimal front (N_{sol}), Riise distance (u) (Yelkenci Kose and Kilincci, 2020), and Zitzler measure ($C_{V,W}$) (Zitzler and Thiele, 1999) were used to evaluate the algorithm convergence performance. The diversity metric (Δ) (Su *et al.*, 2017) and limit distance (τ) (Su *et al.*, 2017) were used to assess the diversity performance of the algorithm. Here, higher N_{sol} and lower u indicate

better convergence. $C_{V,W}$ denotes the percentage of solutions obtained by algorithm V dominated by or equal to at least one solution obtained by algorithm W . The algorithm V is better than algorithm W if $C_{V,W}$ is lower than $C_{W,V}$. A lower Δ indicates better diversity, and a higher τ represents a larger scope of the Pareto optimal solution. For more details on the calculation of u , Δ , and τ . Readers can refer to the literature (Su *et al.*, 2017).

C. Numerical example

Setting

The proposed NSGWA is compared to classical NSGA-II which utilizes the Simulated Binary Crossover (SBX) operator (Cruz *et al.*, 2010) and the mutation operator (Ka. R. B. Deb and Agrawal, 1995) based on Gauss perturbation, $\varepsilon_t \sim N(0, 1)$, because of their efficiency in updating solutions. The coding, initialization operator, and stopping criterion of the proposed algorithms and the NSGA-II are the same. Furthermore, three patterns are tested, and ten replicate calculations run in each pattern because of random initial solutions. Table 7.7 lists the experiment setting.

TAB. 7.7 – Experiment setting for global utilization.

Patterns	N_{ini}	K	I	$\bar{\mu}$
1	10	20	5	(1,1,1,1,1)
2		40	5	
3		20	10	

Results and discussion

The experiment results are listed in table 7.8. Figure 7.14 presents the Pareto optimal front of the NSGWA and NSGA-II. Some conclusions can be drawn as follows:

- (a) u is lower than zero, indicating that the Pareto optimal front obtained by the proposed NSGWA is closer to the Pareto front than that obtained by the NSGA-II. In addition, $C_{1,2}$ is higher than $C_{2,1}$, which presents that the proposed

TAB. 7.8 – Results of the NSGWA and NSGA-II.

Patterns	Algorithms	N_{sol}	u	$C_{1,2}$	$C_{2,1}$	Δ	τ	$t[s]$
1	NSGA-II	13	-7.34	0.46	0.14	0.4899	8.1789	23
	NSGWA	14	\	\	\	0.5109	8.1492	27
2	NSGA-II	14	-9.01	0.64	0.20	0.5680	7.1436	27
	NSGWA	15	\	\	\	0.6090	10.5211	25
3	NSGA-II	12	-11.55	0.66	0	0.5208	11.5187	102
	NSGWA	13	\	\	\	0.4035	15.2015	93

Note: u denotes the Riise distance of the Pareto optimal front of the NSGWA to the NSGA-II; $C_{1,2}$ denotes the Zitzler measure of the NSGA-II to the NSGWA, respectively; $C_{2,1}$ denotes the Zitzler measure of the NSGWA to the NSGA-II.

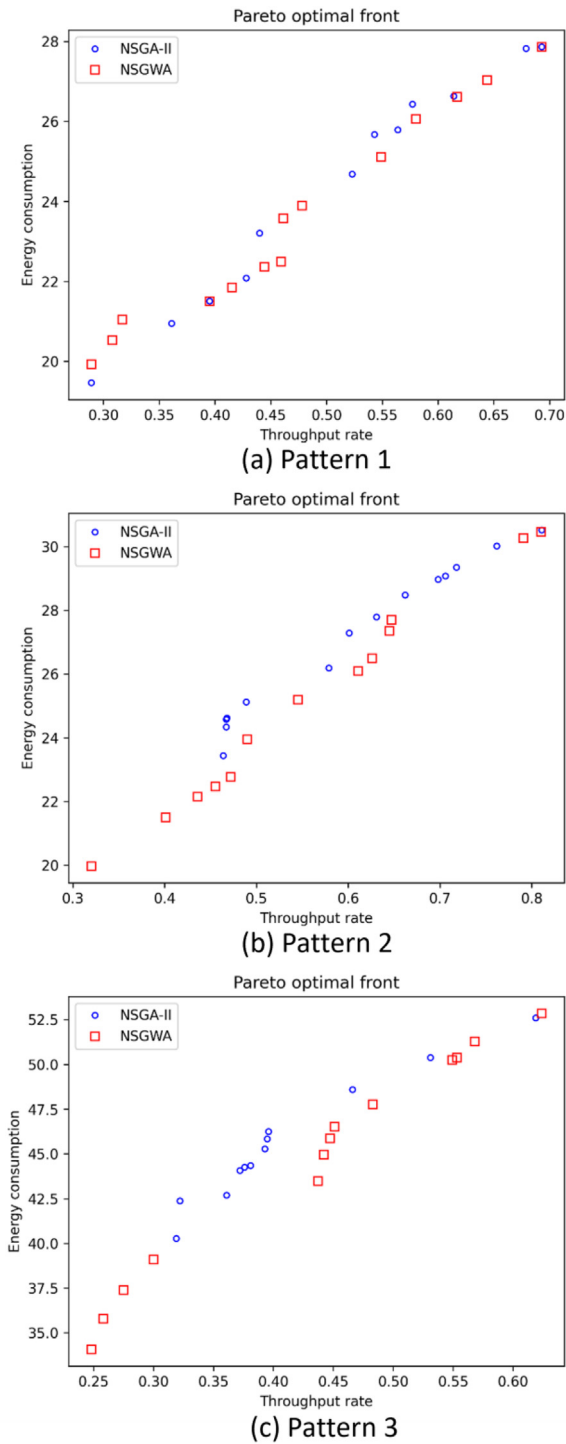


FIG. 7.14 – Pareto optimal front of the NSGWA and NSGA-II.

NSGWA can obtain more nondominated solutions than the NSGA-II. Furthermore, N_{sol} of the proposed NSGWA is more than that of the NSGA-II, indicating the better searchability of the proposed NSGWA. Consequently, the proposed NSGWA achieves better convergence than the NSGA-II.

- (b) Δ of the proposed NSGWA is higher than the NSGA-II in Patterns 1 and 2, indicating the better diversity of the NSGA-II. Because the proposed NSGWA may focus on local search and some solutions are around a small area, its diversity decreases. However, in Pattern 3, with the increase in machine number, the diversity of the proposed NSGWA becomes better than the NSGA-II because the proposed NSGWA can search for more Pareto optimal solutions. In addition, τ of the NSGWA is higher than that of the NSGA-II, indicating the better search scope of the proposed NSGWA. Because the proposed NSGWA balances exploitation and exploration, it can detect more search areas and obtain solutions with a wider scope.
- (c) From Patterns 1 to 3, the candidate solution scope increases. The proposed GWA achieves better convergence and diversity than the NSGA-II, indicating that the proposed NSGWA can perform better in large-scale problems.
- (d) The experimental results show that the proposed NSGWA can achieve better computation efficiency than the NSGA-II.

(2) Local utilization

Unlike global utilization, local utilization considers more about the effect of predictive accuracy on the solution quality of the BAP. Although many machine learning algorithms can build performance prediction models with high predictive accuracy, a large number of predictions is required in the BAP, and errors are accumulated. These may lead to a solution quality loss and local optimal solutions. Consequently, to improve evaluation efficiency without losing solution quality, local utilization of performance prediction integrates generative methods with a simulation or analytical model. The performance prediction replaces part of the performance evaluation based on the simulation or analytical model in the solution process. Figure 7.15 shows the basic flow of global utilization. In this section, the prediction models are integrated with the Degraded Ceiling (DC) (Demir *et al.*, 2019), GWA, and the analytical model built in section 7.2.2 to solve the BAP, presenting the local utilization of performance prediction.

D. Integration of the Degraded Ceiling (DC) algorithm and Gray Wolf Algorithm (GWA) in the BAP

The GWA has been proven to be effective in solving multi-objective BAP. To further improve its local search ability, the DC is integrated into it because of the local search ability of the DC (Li *et al.*, 2016). Furthermore, to improve computation efficiency, the prediction models are used to predict system performance in the local search process. The NSM-II is used to sort solutions. The pseudocode of the Degraded-Ceiling-Gray-Wolf-Algorithm (DC-GWA) is provided. Table 7.9 lists the notion for the DC-GWA.

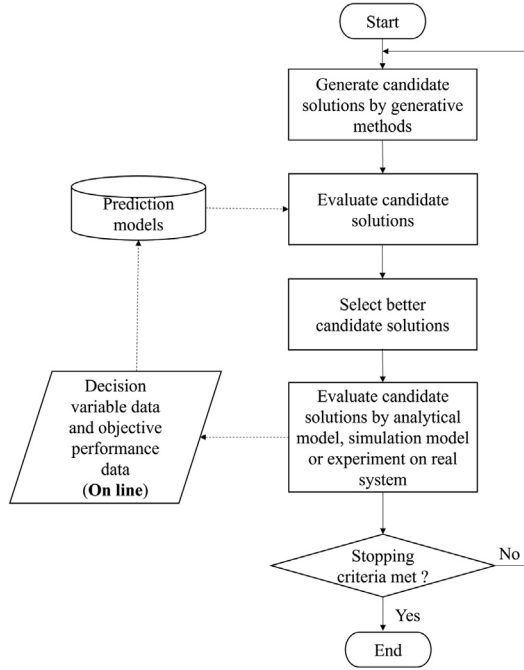


FIG. 7.15 – Basic flow of local utilization for performance prediction.

TAB. 7.9 – Notion in the DC-GWA for the multi-objective BAP.

Items	Description
Π_{arc}	Archive to store current buffer allocation solutions in each iteration
$\Pi_{\text{DC}}, \Pi_{\text{GWA}}$	Archive to store neighborhood buffer allocation solutions
Π_{bes}	Archive to store buffer allocation solutions in the current Pareto optimal front
j_{GWA}	An iteration. J_{GWA} is the maximum number of iterations in the GWA
j_{DC}	An iteration. J_{DC} is the maximum number of iterations in the DC
FL_1	Current Pareto optimal front
N_{ini}	The number of buffer allocation solutions in Π_{arc}
N_{DB}	The number of buffer allocation solutions and their corresponding system performance in the database which is used to build prediction models
i	A machine. I is the number of machines in a manufacturing system
b_i	A buffer capacity. $\bar{b} = (b_1, b_2, \dots, b_{I-1})$
μ_i	A service rate. $\bar{\mu} = (\mu_1, \mu_2, \dots, \mu_{I-1})$
B	Total buffer capacity
θ	Throughput rate
EC	Energy consumption
DL	Ceiling. ΔDL is the increment for the ceiling
$\hat{f}$	Prediction model

Algorithm: the proposed DC-GWA to solve the MBAP

Initialize:

Generate N_{DB} initial solutions and their corresponding throughput and energy consumption in the database DB;

Randomly select N_{ini} initial solutions in DB to build archive Π_{arc} .

1	Set $j_{GWA} = 0$
2	while $j_{GWA} < J_{GWA}$ do
3	$j_{GWA} = j_{GWA} + 1$
4	Set $j_{DC} = 0$.
5	Build prediction models f based on DB.
6	Set $\Pi_{DC} = \Pi_{arc}$.
7	Initialize DL.
8	while $j_{DC} < J_{DC}$ do
9	$j_{DC} = j_{DC} + 1$
10	Generate neighborhood solutions of solutions in Π_{DC} using equations (7.26)–(7.28).
11	Accept the neighborhood solutions based on the acceptance criterion.
12	Predict θ and EC using f .
13	Give floor rank and calculate the crowding distance of the accepted neighborhood solutions.
14	Update Π_{DC} by the solutions in FL_1 .
15	$DL = DL + \Delta DL$
16	end while
17	Generate neighborhood solutions of solutions in Π_{arc} and Π_{DC} using equations (7.19)–(7.23), and store them in Π_{GWA} .
18	Revise buffer capacities in the neighborhood solutions by equation (7.24).
19	Normalize the neighborhood solutions by equation (7.25).
20	Calculate θ and EC using the analytical model built in section 7.2.2.
21	Give floor rank and calculate the crowding distance of the solutions in Π_{arc} , Π_{DC} , and Π_{GWA} .
22	Update Π_{bes} by the solutions in FL_1 .
23	Select N_{ini} solutions and update Π_{arc} based on the NSM-II.
24	Update DB by adding the solutions in Π_{bes} into DB and delete the same amount of solutions.
25	end while
26	Output Π_{bes} .

Genetic operator

The operators to update candidate buffer allocation solutions in the DC are as follows:

$$b_i(j_{DC} + 1) = b_i(j_{DC}) - \Delta b, \quad b_i(j_{DC} + 1) > 0 \quad (7.26)$$

$$b_k(j_{\text{DC}} + 1) = b_k(j_{\text{DC}}) + \Delta b, \quad k \neq i \quad (7.27)$$

$$\Delta b = \text{randint}(1, 2) \quad (7.28)$$

where $b_i(j_{\text{DC}})$ denotes the capacity of buffer i in iteration j_{DC} ; $b_k(j_{\text{DC}})$ denotes the capacity of buffer k in iteration j_{DC} ; Δb denotes a capacity increment and it is equal to 1 or 2; $\text{randint}(1, 2)$ denotes 1 or 2.

As for the operators in the GWA, equations (7.19)–(7.23) are used.

Acceptance criterion

A neighborhood solution in the DC is accepted if the following condition is met:

$$\text{DL} < \frac{\theta_{\text{nei}} - \theta_{\text{ori}}}{\theta_{\text{ori}}} + \frac{\text{EC}_{\text{ori}} - \text{EC}_{\text{nei}}}{\text{EC}_{\text{ori}}} \quad (7.29)$$

where θ_{ori} and θ_{nei} denote the throughput rate corresponding to a selected solution and its neighborhood solution, respectively; EC_{ori} and EC_{nei} denote the energy consumption corresponding to a selected solution and its neighborhood solution, respectively.

Floor rank and crowding distance calculation

This section also uses the NSM-II to sort candidate buffer allocation solutions.

Stopping criterion

The stopping criterion for the DC and GWA is defined as the maximum number of iterations (J_{DC} and J_{GWA}), respectively.

E. Performance metrics

This section also uses the number of solutions in the Pareto optimal front (N_{sol}), Riise distance (u), and Zitzler measure ($C_{V,W}$) to evaluate the algorithm convergence performance. The diversity metric (Δ) and limit distance (τ) are used to assess the diversity performance of the algorithm.

F. Numerical example

Setting

The proposed DC-GWA is compared to classical NSGA-II because of its efficiency in updating solutions. Table 7.10 lists the parameter settings of the proposed DC-GWA and NSGA-II. Furthermore, three patterns are tested, and ten replicate calculations run in each pattern because of random initial solutions. Table 7.11 lists the experiment setting.

Results and discussion

The experiment results are listed in table 7.12. Figure 7.16 presents the Pareto optimal front of the DC-GWA and NSGA-II. Some conclusions can be drawn as follows:

- (a) The experimental results demonstrate a better convergence of the proposed DC-GWA than the NSGA-II. u is lower than zero, indicating that the Pareto optimal front obtained by the proposed DC-GWA is closer to the Pareto front than that obtained by the NSGA-II. Furthermore, $C_{1,2}$ is higher than $C_{2,1}$,

TAB. 7.10 – Parameter setting.

Algorithms	Items	Setting
NSGA-II	Initialization operator	Randomly generate initial solutions
	Crossover operation	Utilizes the Simulated Binary Crossover (SBX) operator (Cruz <i>et al.</i> , 2010)
	Mutation operation	Utilize the mutation operator (Ka. R. B. Deb and Agrawal, 1995) based on Gauss perturbation, $\varepsilon_i \sim N(0, 1)$
	Sorting method	NSM-II
	Stopping criterion	Maximum iteration
	Initial solution size	10
	Crossover rate	0.85
	Mutation rate	0.03
	Maximum iteration	500
DC-GWA	Initialization operator	Randomly generate initial solutions
	Stopping criterion	Maximum iteration of the GWA
	Initial solution size	10
	Maximum iteration of the DC	10
	Maximum iteration of the GWA	50
	Initial ceiling	0.001
	Ceiling increment	0.0001

TAB. 7.11 – Experiment setting for local utilization.

Patterns	N_{ini}	K	I	$\bar{\mu}$
1	10	40	10	(1,1,1,1,1,1,1,1,1)
2		60	15	
3		80	20	

TAB. 7.12 – Results of the DC-GWA and NSGA-II.

Patterns	Algorithms	N_{sol}	u	$C_{1,2}$	$C_{2,1}$	Δ	τ	$t[s]$
1	NSGA-II	13	-40.39	0.93	0.03	0.5327	13.9102	76
	DC-GWA	32	\	\	\	0.6908	16.9971	62
2	NSGA-II	12	-54.21	0.86	0	0.5154	11.0941	154
	DC-GWA	30	\	\	\	0.6798	18.1286	148
3	NSGA-II	11	-67.45	0.72	0	0.4987	15.0981	399
	DC-GWA	20	\	\	\	0.7012	26.9876	501

Note: u denotes the Riise distance of the Pareto optimal front of the DC-GWA to the NSGA-II; $C_{1,2}$ denotes the Zitzler measure of the NSGA-II to the DC-GWA, respectively; $C_{2,1}$ denotes the Zitzler measure of the DC-GWA to the NSGA-II.

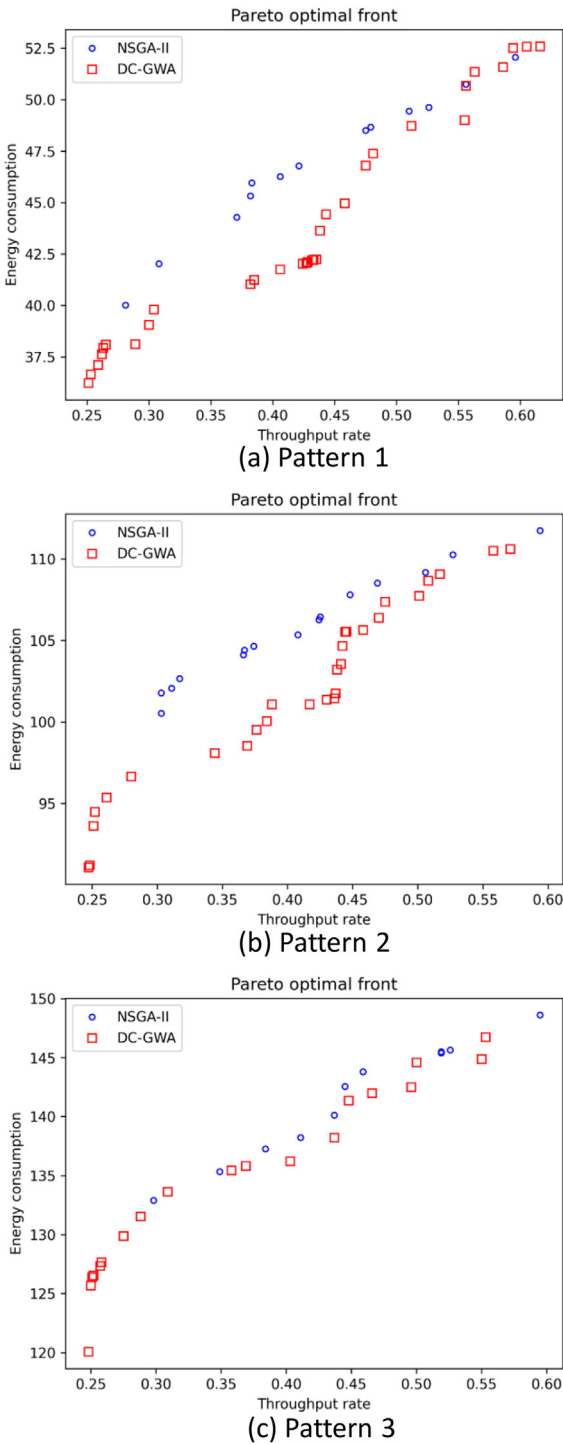


FIG. 7.16 – Pareto optimal front of the DC-GWA and NSGA-II.

which presents that the proposed DC-GWA can obtain more nondominated solutions than the NSGA-II. Furthermore, the proposed DC-GWA has better global searchability than the NSGA-II because it obtains far more N_{sol} than that obtained by the NSGA-II. In addition, with the total buffer capacity and machine number increasing from Patterns 1 to 3, the solution scope is enlarged increasingly. The proposed DC-GWA still outperforms the NSGA-II, and the difference between the two-algorithm increase. These demonstrate the performance of the proposed DC-GWA in large-scale problems.

- (b) Δ of the proposed DC-GWA is higher than the NSGA-II because the proposed DC-GWA focuses on local search, and some solutions are around a small region. However, because of its strong searchability, the Pareto optimal solutions are relatively uniformly distributed in figure 7.16. Therefore, its diversity is still acceptable. In addition, τ of the DC-GWA is far higher than that of the NSGA-II, indicating its better search scope. This results from the balance between exploitation and exploration in the DC-GWA which obtains solutions with a wider scope.
- (c) In small-scale problems such as Patterns 1 and 2, the proposed DC-GWA has a better computation efficiency than the NSGA-II. However, in Pattern 3 which belongs to a large-scale problem, the computation time of the proposed DC-GWA becomes longer than that of the NSGA-II. This results from the solution scope increase where the proposed DC-GWA can detect more candidate solutions.

7.4.2 The Other Application

Figure 7.17 shows the potential application of the prediction models in intelligent manufacturing systems. Besides system configuration, prediction models can also be used to solve the following problems in the process control of intelligent manufacturing systems:

- (a) Maintenance planning: faults may occur during the operation of an intelligent manufacturing system. Maintenance planning is necessary to deal with the faults properly and decrease their effects on the production rate. One option is to integrate performance prediction with optimization algorithms to solve the maintenance planning problem. Performance prediction efficiently evaluates candidate maintenance strategies, while optimization algorithms keep generating candidate maintenance strategies.
- (b) Fault detection: machine faults, processing errors, and unreasonable scheduling significantly fluctuate system performance. The differences can be analyzed by comparing the data between the normal operating condition and fault condition of a system. Then the fault prediction models can be built using machine learning algorithms to predict the fault probability.
- (c) Bottleneck detection: it is well-known that bottlenecks constrain the productivity of an intelligent manufacturing system. The bottlenecks refer to the machine with the lowest performance. The prediction of the bottlenecks of future production runs provides a chance to reallocate system resources

reasonably. Consequently, based on historical data of machines in a system, bottleneck prediction models can be built to predict system bottlenecks.

- (d) Remaining useful life assessment: the performance of an intelligent manufacturing system degrades with the increase of its use time. Performance degradation prediction is vital for assessing the remaining useful life and making decisions for maintenance. The performance degradation prediction models are typically built based on the data of the system's normal operating condition.

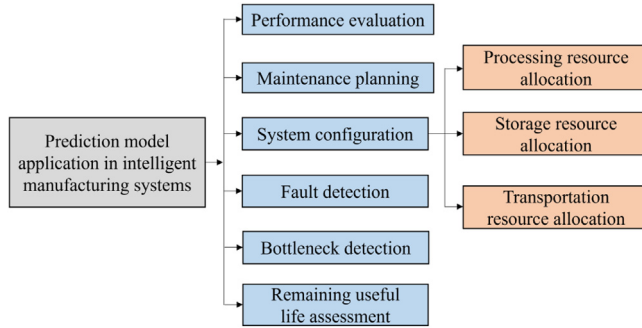


FIG. 7.17 – Potential application of the prediction models.

7.5 Conclusions

This chapter introduced two kinds of prediction models for intelligent manufacturing systems, including the regression algorithm-based prediction model and the artificial neural network-based prediction model. Two methods, which are data-driven and model-driven methods, are utilized to generate data for building prediction models. The numerical examples are conducted. Experimental results show that the regression algorithms outperform the ANN in the performance prediction of intelligent manufacturing systems. Furthermore, the performance prediction models can be integrated with optimization algorithms to configure resources of intelligent manufacturing systems. According to utilization, the prediction model can be used globally or locally. Two algorithms, which are NSGWA and DC-GWA, are proposed to show the global and local utilization of the prediction models. Experimental results indicate the effectiveness of the prediction models in system configuration. In addition, the proposed NSGWA and DC-GWA outperform the classical NSGA-II. This phenomenon results from the better searchability of the two proposed algorithms.

The potential applications of the prediction models for intelligent manufacturing systems are analyzed at last. The performance prediction models can be integrated with optimization algorithms to configure system resources and plan maintenance schedules. In addition, they can also be used in the field of fault detection, bottleneck detection, and remaining useful life assessment.

References

- Agrawal R. B., Deb K., Agrawal R. B. (1994) Simulated binary crossover for continuous search space, *Complex Syst.* **9**(3), 115–148.
- Alaouchiche Y., Ouazene Y., Yalaoui F. (2020) Economic and energetic performance evaluation of unreliable production lines: An integrated analytical approach, *IEEE Access* **8**, 185330–185345.
- Cruz F. R. B., Van Woensel T., Smith J. M. (2010) Buffer and throughput trade-offs in M/G/1/K queueing networks: A bi-criteria approach, *Int. J. Prod. Econ.* **125**(2), 224–234.
- Deb K., Agrawal R. B. (1995) Simulated binary crossover for continuous search space, *Complex Syst.* **9**(2), 115–148.
- Deb K., Pratap A., Agarwal S., Meyarivan T. (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.* **6**(2), 182–197, Article Pii s 1089-778x(02) 04101-2.
- Demir L., Diamantidis A. C., Eliyi D. T., O’Kelly M. E., Tunali S. (2019) Optimal buffer allocation for serial production lines using heuristic search algorithms: A comparative study, *Int. J. Ind. Syst. Eng.* **33**(2), 252.
- Demir L., Koyuncuoglu M. U. (2021) The impact of the optimal buffer configuration on production line efficiency: A VNS-based solution approach, *Expert Syst. Appl.* **172**, Article 114631.
- Demir L., Tunali S., Eliyi D. T. (2014) The state of the art on buffer allocation problem: A comprehensive survey, *J. Intell. Manuf.* **25**(3), 371–392.
- Dereje G., Abeya G., Gopal M. (2020) Manufacturing system modeling and performance analysis of mineral water production line using ARENA simulation, *Int. J. Eng. Adv. Technol.* **9**(5), 2249–8958.
- Gao S., Liu H. (2023) A data-driven ensemble algorithm of black widow optimizer and simulated annealing algorithms for multi-objective buffer allocation in production lines, *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* <https://doi.org/10.1177/09544054231157153>.
- Gao S. X., Kobayashi T., Tajiri A., Ota J. (2021) Throughput analysis of conveyor systems involving multiple materials based on capability decomposition, *Comput. Ind.* **132**, Article 103526.
- Geurts P., Ernst D., Wehenkel L. (2006) Extremely randomized trees, *Mach. Learn.* **63**(1), 3–42.
- Gordon W. J., Newell G. F. (1967) Cyclic queueing systems with restricted length queues, *Oper. Res.* **15**(2), 266–277.
- Got A., Moussaoui A., Zouache D. (2020) A guided population archive whale optimization algorithm for solving multiobjective optimization problems, *Expert Syst. Appl.* **141**, Article 112972.
- Gu X., Guo W. H., Jin X. N. (2020) Performance evaluation for manufacturing systems under control-limit maintenance policy, *J. Manuf. Syst.* **55**, 221–232.
- Huang C. L., Huang Y. H., Chang T. Y., Chang S. H., Chung C. H., Huang D. T., Li R. K. (1999) The construction of production performance prediction system for semiconductor manufacturing with artificial neural networks, *Int. J. Prod. Res.* **37**(6), 1387–1402.
- Ke G. L., Meng Q., Finley T., Wang T. F., Chen W., Ma W. D., Ye Q. W., Liu T. Y. (2017) *LightGBM: A Highly Efficient Gradient Boosting Decision Tree*. Advances in Neural Information Processing Systems 30 (NIPS 2017).
- Kose S. Y., Kilincci O. (2020) A multi-objective hybrid evolutionary approach for buffer allocation in open serial production lines, *J. Intell. Manuf.* **31**(1), 33–51.
- Li L., Qian Y. L., Yang Y. M., Du K. (2016) A fast algorithm for buffer allocation problem, *Int. J. Prod. Res.* **54**(11), 3243–3255.
- Mirjalili S., Mirjalili S. M., Lewis A. (2014) Grey wolf optimizer, *Adv. Eng. Softw.* **69**, 46–61.
- Mirjalili S., Saremi S., Mirjalili S. M., Coelho L. D. (2016) Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization, *Expert Syst. Appl.* **47**, 106–119.
- Mohammad F. A., Ismail S. Z. (2021) Product-Service System (PSS) for Inventory Control: Analysis Using Delmia QUEST Simulation Modelling Based on the Production Line.
- Nadimi-Shahraki M. H., Taghian S., Mirjalili S. (2021). An improved grey wolf optimizer for solving engineering problems, *Expert Syst. Appl.* **166**, Article 113917.

- Oljira D. G., Abeya T. G., Ofgera G., Gopal M. (2020) Manufacturing system modeling and performance analysis of mineral water production line using ARENA simulation, *Int. J. Eng. Adv. Technol.* **9**(5), 312–317.
- Sadr J., Malhame R. P. (2004) Decomposition/aggregation-based dynamic programming optimization of partially homogeneous unreliable transfer lines, *IEEE Trans. Automatic Control* **49**(1), 68–81.
- Shin J., Grosbard D., Morrison J. R., Kalir A. (2019) Decomposition without aggregation for performance approximation in queueing network models of semiconductor manufacturing, *Int. J. Prod. Res.* **57**(22), 7032–7045.
- Si M. X., Du K. (2020) Development of a predictive emissions model using a gradient boosting machine learning method, *Environ. Technol. Innov.* **20**, Article 101028.
- Song Z. Y., Moon Y. (2019) Performance analysis of cybermanufacturing systems, *Proc. Inst. Mech. Eng. Part B J. Eng. Manuf.* **233**(5), 1362–1376.
- Su C., Shi Y. M., Dou J. P. (2017) Multi-objective optimization of buffer allocation for remanufacturing system based on TS-NSGAI hybrid algorithm, *J. Clean. Prod.* **166**, 756–770.
- Tsadiras A. K., Papadopoulos C. T., O’Kelly M. E. J. (2013) An artificial neural network based decision support system for solving the buffer allocation problem in reliable production lines, *Comput. Ind. Eng.* **66**(4), 1150–1162.
- Wang J. L., Xu C. Q., Zhang J., Zhong R. (2022) Big data analytics for intelligent manufacturing systems: A review, *J. Manuf. Syst.* **62**, 738–752.
- Weiss S., Schwarz J. A., Stolletz R. (2019) The buffer allocation problem in production lines: Formulations, solution methods, and instances, *IIE Trans.* **51**(5), 456–485.
- Xi S. H., Smith J. M., Chen Q. X., Mao N., Zhang H. Y., Yu A. L. (2022) Simultaneous machine selection and buffer allocation in large unbalanced series-parallel production lines, *Int. J. Prod. Res.* **60**(7), 2103–2125.
- Yan F. Y., Wang J. Q., Li Y., Cui P. H. (2021) An improved aggregation method for performance analysis of Bernoulli serial production lines, *IEEE Trans. Autom. Sci. Eng.* **18**(1), 114–121.
- Zitzler E., Thiele L. (1999) Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach, *IEEE Trans. Evol. Comput.* **3**(4), 257–271.

